Celso Carneiro Ribeiro

Introdução aos Modelos e Métodos de Otimização em Pesquisa Operacional

Parte II – Programação Inteira

Operador Nacional do Sistema Elétrico

2004

Parte II – Programação inteira

- Origens
- Motivação: PL vs. PI
- Solução gráfica
- Formulação de problemas
- Métodos de cortes (princípios básicos)
- Métodos de branch-and-bound
- Aplicação
- Teoria da complexidade
- Laboratório de programação inteira

Origens

- Primeira tentativa de resolver o problema do caixeiro viajante (Dantzig, 1954): 49 cidades (Simplex + cortes)
- Gomory, anos 60: métodos de cortes
- Land, Doig e Roy, anos 60: métodos de "branch-and-bound"
- Dificuldades numéricas e computacionais: "desilusão"
- Teoria da complexidade, anos 70-80: problemas NP-completos, necessidade e progressos em heurísticas
- Ressurgimento, anos 80-90: métodos poliedrais
- Maior TSP resolvido atualmente: 15.112 cidades

Bibliografia

- T. Hu, "Integer Programming and Network Flows", 1969
- H. Salkin, "Integer Programming", 1975
- C. Papadimitriou e K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity", 1982
- G. Nemhauser e L. Wolsey, "Integer and Combinatorial Optimization", 1988
- W. Cook, W. Cunningham, W. Pulleyblank e A. Schrijver, "Combinatorial Optimization", 1998
- L. Wolsey, "Integer Programming", 1998

Motivação

- Problemas de programação linear:
  - Função objetivo linear
  - Restrições lineares
    - igualdades
    - desigualdades
  - Variáveis contínuas (reais)
- Problemas de programação (linear) inteira:
  - Variáveis assumem apenas valores inteiros

Motivação

- Problema das ligas metálicas: Uma metalúrgica deseja maximizar sua receita bruta. A tabela a seguir ilustra a proporção de cada material na mistura para a obtenção das ligas passíveis de fabricação, assim como a disponibilidade de cada matéria prima (em toneladas) e os preços de venda por tonelada de cada liga. Qual deve ser a quantidade produzida de cada liga?

Motivação

	Liga 1	Liga 2	Disponibilidade
Cobre	0,5	0,2	16 ton.
Zinco	0,25	0,3	11 ton.
Chumbo	0,25	0,5	15 ton.
Preço	3.000	5.000	

Motivação

- Variáveis de decisão:

  x_1: quantidade produzida da liga 1 (em toneladas)

  x_2: quantidade produzida da liga 2 (em toneladas)

- As variáveis de decisão representam quantidades contínuas.

Motivação

- Função objetivo:

  maximizar $3.000\ x_1 + 5.000\ x_2$

- Disponibilidade das matérias primas:

  $0{,}5\ x_1 + 0{,}2\ x_2 \leq 16$ cobre

  $0{,}25\ x_1 + 0{,}3\ x_2 \leq 11$ zinco

  $0{,}25\ x_1 + 0{,}5\ x_2 \leq 15$ chumbo

- Todas as quantidades produzidas são não-negativas:

  $x_1, x_2 \geq 0$

Motivação

- Problema de produção de fósforos: Uma empresa produz dois tipos de fósforos. Esta empresa tem um lucro de 3 (x100 u.m.) em cada caixa de fósforos longos e de 2 (x100 u.m.) em cada caixa de fósforos curtos. Fósforos dos dois tipos são feitos por uma única máquina, que pode produzir 9 (x100.000) caixas de fósforos por ano. Para produzir e vender os fósforos, a empresa precisa de madeira e de caixas. São necessários 3 m^3 de madeira para cada caixa de fósforos longos e 1 m^3 de madeira para cada caixa de fósforos curtos.

Motivação

A empresa possui 18 (x100.000) m^3 de madeira para usar durante o próximo ano. Dispõe ainda de 7 (x100.000) caixas para fósforos longos e 6 (x100.000) caixas para fósforos curtos. A empresa deseja maximizar seus lucros com a venda de fósforos no próximo ano, sabendo que toda sua produção pode ser vendida.

- Variáveis de decisão:

  x_1: número de caixas (x100.000) de fósforos longos

  x_2: número de caixas (x100.000) de fósforos curtos

Motivação

- Restrições:

  Capacidade máxima anual de produção da máquina:

  $$x_1 + x_2 \leq 9 \qquad \text{(x100.000)}$$

  Quantidade de madeira disponível:

  $$3x_1 + x_2 \leq 18 \qquad \text{(x100.000)}$$

  Número máximo de caixas disponíveis:

  $$x_1 \leq 7 \qquad \text{(x100.000)}$$

  $$x_2 \leq 6 \qquad \text{(x100.000)}$$

  Não-negatividade: $x_1 \geq 0$ e $x_2 \geq 0$

Motivação

- Problema de programação linear:

maximizar $3x_1 + 2x_2$

sujeito a :

$x_1 + x_2 \leq 9$ restrição (1)

$3x_1 + x_2 \leq 18$ restrição (2)

$x_1 \leq 7$ restrição (3)

$x_2 \leq 6$ restrição (4)

$x_1, x_2 \geq 0$

- As variáveis deste problema são realmente contínuas?
- Aproximação realista

Motivação

- Problema de escalonamento de horários: Um hospital deseja planejar os horários das enfermeiras de seu turno da noite. A demanda por enfermeiras no turno da noite no dia j=1,...,7 da semana é um número inteiro d_j. Cada enfermeira trabalha cinco dias consecutivos e descansa nos dois dias seguintes. O objetivo consiste em minimizar o número de enfermeiras contratadas.
- Variáveis de decisão:

  x_j: número de enfermeiras que começam seu horário no dia j=1,...,7

Motivação

- Modelo:

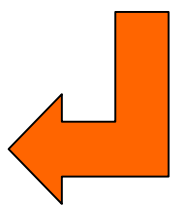
minimizar $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$

sujeito a :

$$x_1 + x_4 + x_5 + x_6 + x_7 \geq d_1$$
$$x_1 + x_2 + x_5 + x_6 + x_7 \geq d_2$$
$$x_1 + x_2 + x_3 + x_6 + x_7 \geq d_3$$
$$x_1 + x_2 + x_3 + x_4 + x_7 \geq d_4$$
$$x_1 + x_2 + x_3 + x_4 + x_5 \geq d_5$$
$$x_2 + x_3 + x_4 + x_5 + x_6 \geq d_6$$
$$x_3 + x_4 + x_5 + x_6 + x_7 \geq d_7$$
$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0$$
$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \text{ inteiros}$$

Problema de programação linear inteira: seu valor ótimo é superior ao do problema linear associado, pois contém restrições adicionais (todas as variáveis devem assumir valores inteiros).

O número de enfermeiras **nunca** pode ser fracionário!



Motivação

- Problema da mochila: Um viajante dispõe de n itens que deve selecionar para colocar em uma mochila que está sendo preparada para uma viagem. O peso do item j é igual a_j e o "lucro" obtido caso ele seja selecionado e colocado na mochila é igual a c_j, para j=1,...,n. Quais itens devem ser selecionados, sabendo-se que o peso máximo que o viajante pode carregar na mochila é igual a b?
- Variáveis de decisão:

  x_j: quantidade selecionada do item j

Motivação

- Caso (1): os itens podem ser fracionados e não há limite na quantidade selecionada

$$\text{maximizar} \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeito a}: \sum_{j=1}^{n} a_j x_j \leq b$$

$$x_j \geq 0, \quad j = 1,\ldots,n$$

- Problema de programação linear
- Solução trivial: selecionar o item j* cuja razão c_j/a_j é máxima e fazer $x_{j*}=b/a_{j*}$, $x_j = 0$ para os demais.

Motivação

- Caso (2): os itens não podem ser fracionados e não há limite na quantidade selecionada

$$\text{maximizar} \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeito a:} \sum_{j=1}^{n} a_j x_j \leq b$$

$$x_j \geq 0 \text{ e inteiro}, \quad j = 1,\ldots,n$$

- Problema de programação inteira
- Solução não trivial!
- Neste caso, as variáveis inteiras indicam o número de objetos de cada tipo que serão selecionados.

Motivação

- Caso (3): os itens podem ser fracionados e no máximo uma unidade de cada item pode ser selecionada

$$\text{maximizar} \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeito a}: \sum_{j=1}^{n} a_j x_j \leq b$$

$$1 \geq x_j \geq 0, \quad j = 1,\ldots,n$$

- Problema de programação linear
- Solução trivial: ordenar os itens pela razão c_j/a_j e fazer $x_j=1$ enquanto couber, fracionar o objeto seguinte e fazer $x_j = 0$ para os demais.

Motivação

- Exemplo:

$$\text{maximizar } 6x_1 + 8x_2 + 4x_3 + x_4$$

$$\text{sujeito a : } x_1 + 2x_2 + 2x_3 + x_4 \leq 4$$

$$1 \geq x_j \geq 0, \qquad j = 1,2,3,4$$

- Maior razão: $c_1/a_1 = 6/1 = 6 \Rightarrow x_1 = 1$
- Segunda maior razão: $c_2/a_2 = 8/2 = 4 \Rightarrow x_2 = 1$
- Terceira maior razão: $c_3/a_3 = 4/2 = 2 \Rightarrow x_3 = 0,5$
- $x_4 = 0$

Motivação

- Caso (4): os itens não podem ser fracionados e no máximo uma unidade de cada item pode ser selecionada

$$\text{maximizar} \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeito a}: \sum_{j=1}^{n} a_j x_j \leq b$$

$$x_j \in \{0,1\}, \quad j = 1,\ldots,n$$

- Problema de programação inteira
- Solução não trivial!
- Neste caso, as variáveis inteiras (binárias ou 0-1) representam a decisão de selecionar um objeto ou não.

Motivação

- Exemplo:

$$\text{maximizar } 12x_1 + 7x_2 + 6x_3$$

$$\text{sujeito a : } 3x_1 + 2x_2 + 2x_3 \leq 4$$

$$x_j \in \{0,1\}, \qquad j = 1,2,3$$

- Maior razão: $c_1/a_1 = 12/3 = 4 \Rightarrow x_1 = 1$
- $x_2 = x_3 = 0$
- Esta solução é ótima?

  Não, o problema de programação inteira é mais difícil!!!

  (lucro = 12)
- Solução ótima: $x_1 = 0$, $x_2 = 1$, $x_3 = 1$ (lucro = 13)

Motivação

- Uma empresa de laticínios produz queijos e leite. Parte do transporte da produção é feito pela própria empresa, enquanto o restante é terceirizado. O atual parque de veículos da empresa está obsoleto e será modernizado. Dois tipos de novos veículos são considerados para a substituição dos atuais. Apenas queijos podem ser transportados através de veículos do tipo A, em um total de no máximo 100 (x100 kg) por mês. Queijos e leite podem ser transportados por veículos do tipo B, em um total mensal de no máximo 50 (x100 kg) de queijo e 20 (x 100 l) de leite.

Motivação

A compra de um veículo do tipo A proporciona uma economia mensal de 1000 (x1 $) em relação à contratação externa da distribuição. Para um veículo do tipo B, a economia mensal é de 700 (x1 $). A empresa deseja maximizar suas economias mensais. A demanda diária mínima a ser transportada é de 2425 (x100 kg) de queijo e de 510 (x100 l) de leite. De modo a evitar investimentos em capacidade ociosa, foi determinado que a capacidade da frota não deve exceder a demanda diária mínima. Determinar o número de veículos de cada tipo que devem ser adquiridos.

Motivação

- Variáveis de decisão:

  x_1: número de veículos do tipo A adquiridos

  x_2: número de veículos do tipo B adquiridos

- Como o número de veículos não pode ser fracionário:

  $x_1 \geq 0$ e inteiro

  $x_2 \geq 0$ e inteiro

Motivação

- Restrições sobre o número máximo de veículos adquiridos:

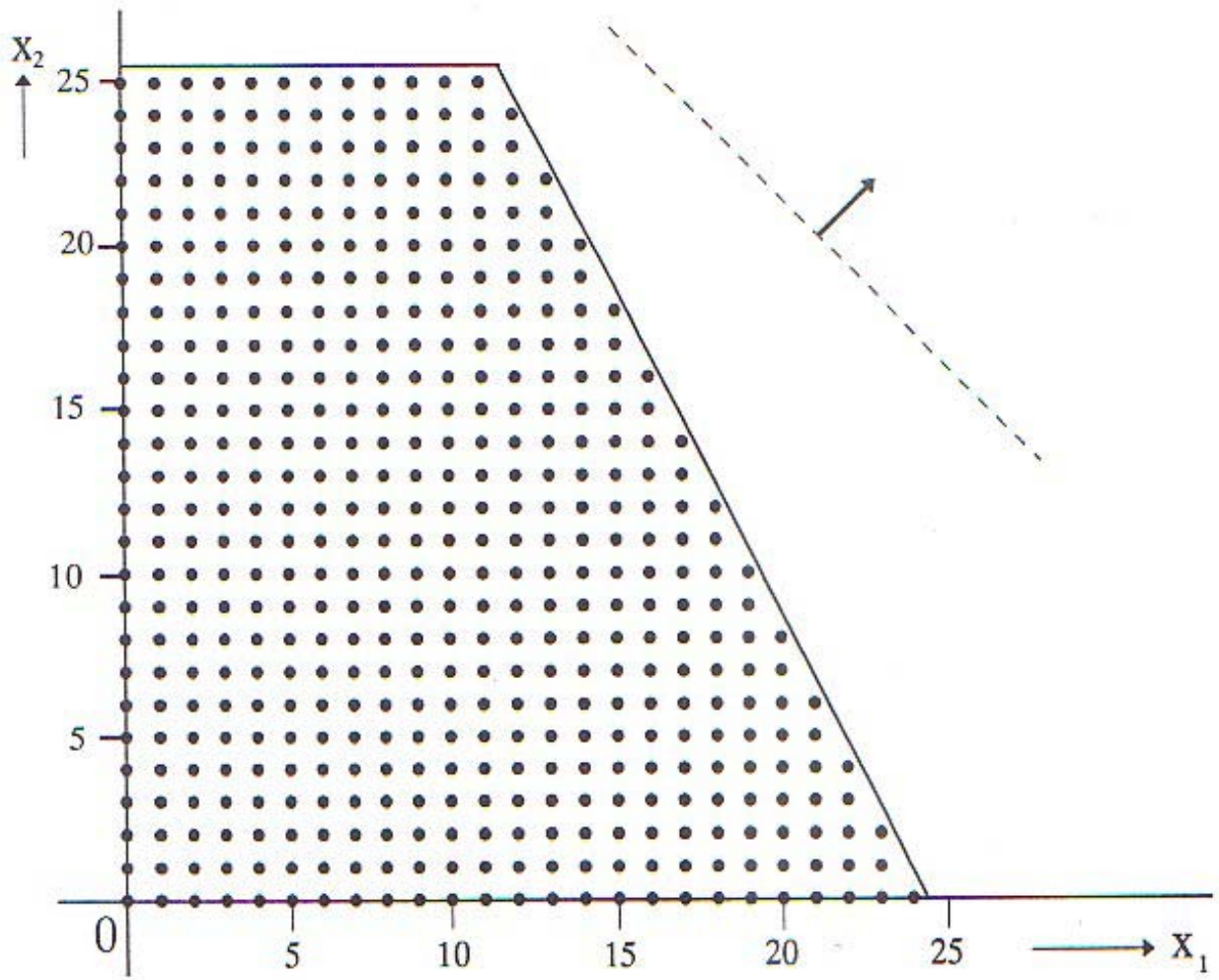
$$100\, x_1 + 50\, x_2 \leq 2425$$

$$20\, x_2 \leq 510$$

- Função objetivo: maximização das economias

maximizar $1000\, x_1 + 700\, x_2$

Motivação



maximizar $1000x_1 + 700x_2$

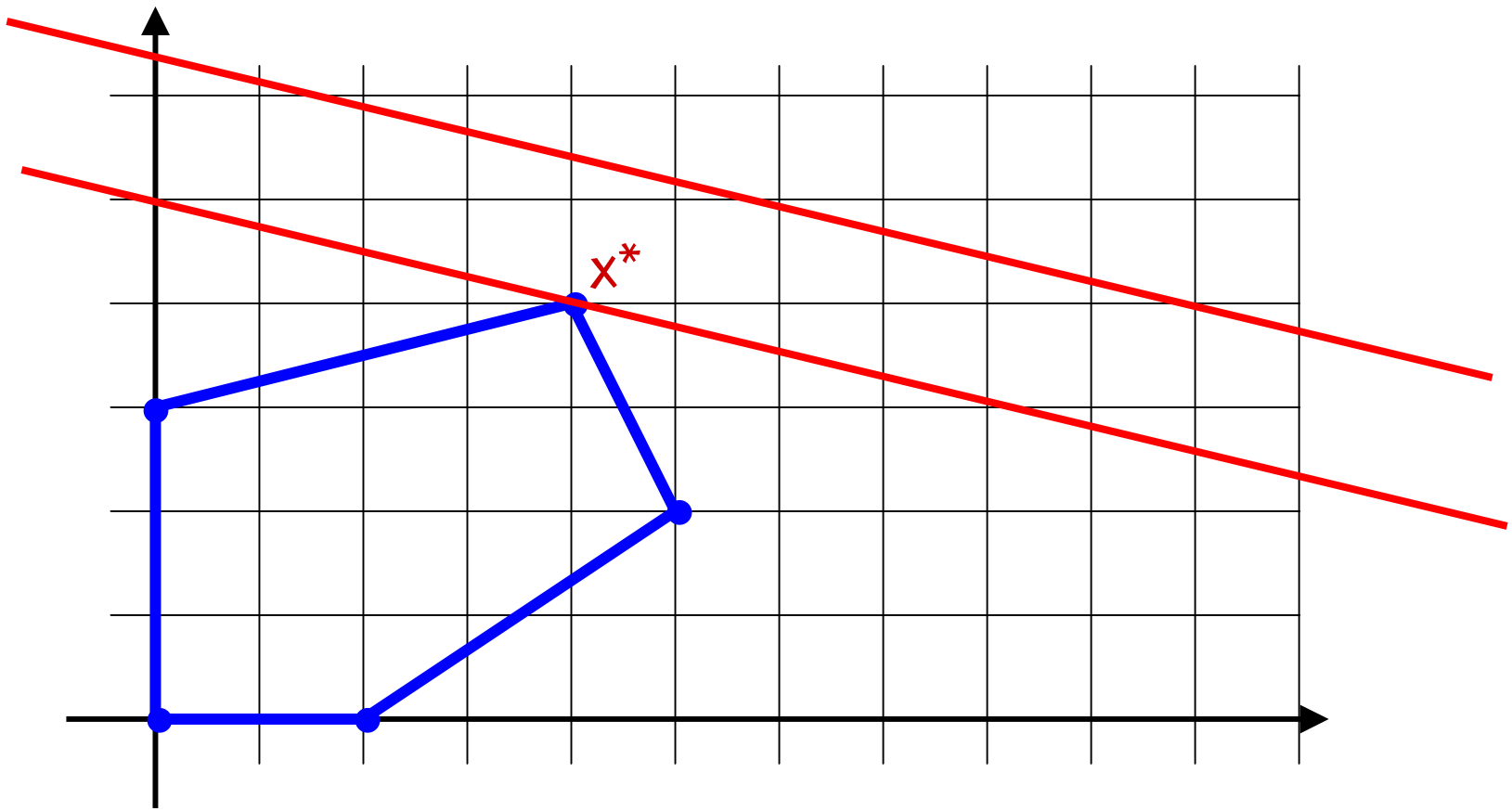
sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros

Solução gráfica

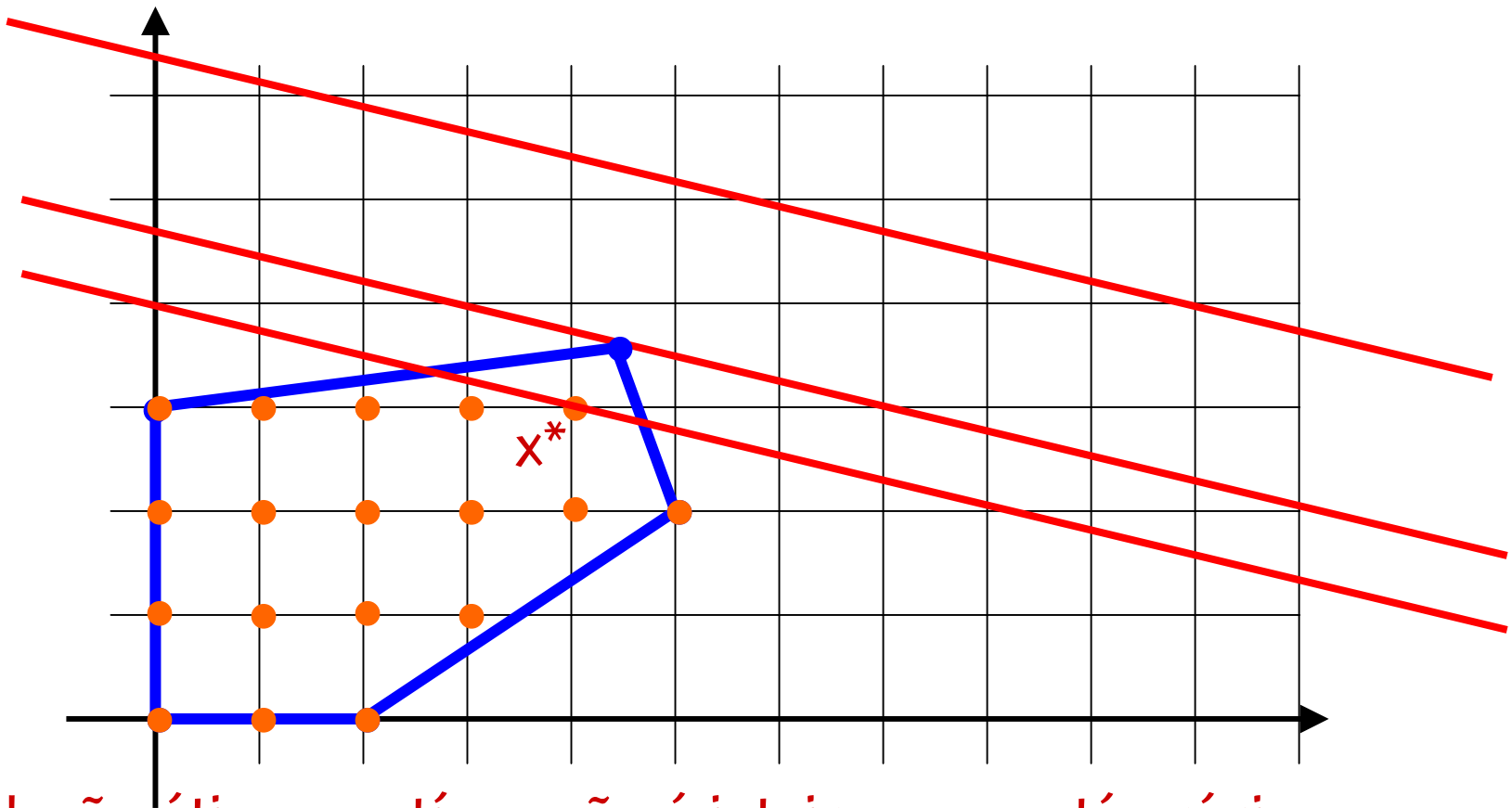
Caso 1:



Solução ótima contínua é inteira $\Rightarrow$ também é ótima inteira

Solução gráfica

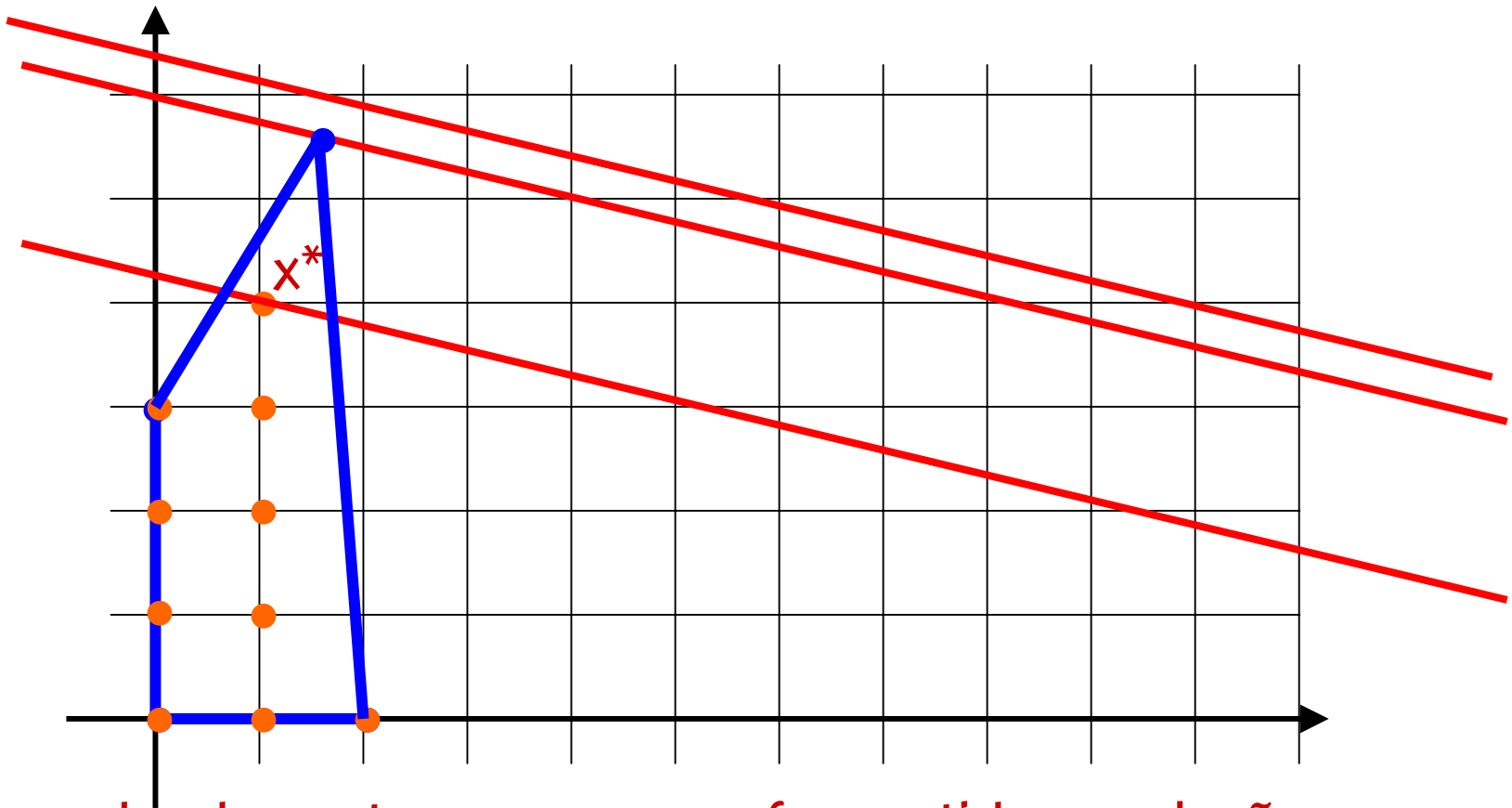
Caso 2:



Solução ótima contínua não é inteira, mas está próxima.
O ótimo inteiro pode ser obtido por arredondamento.

Solução gráfica

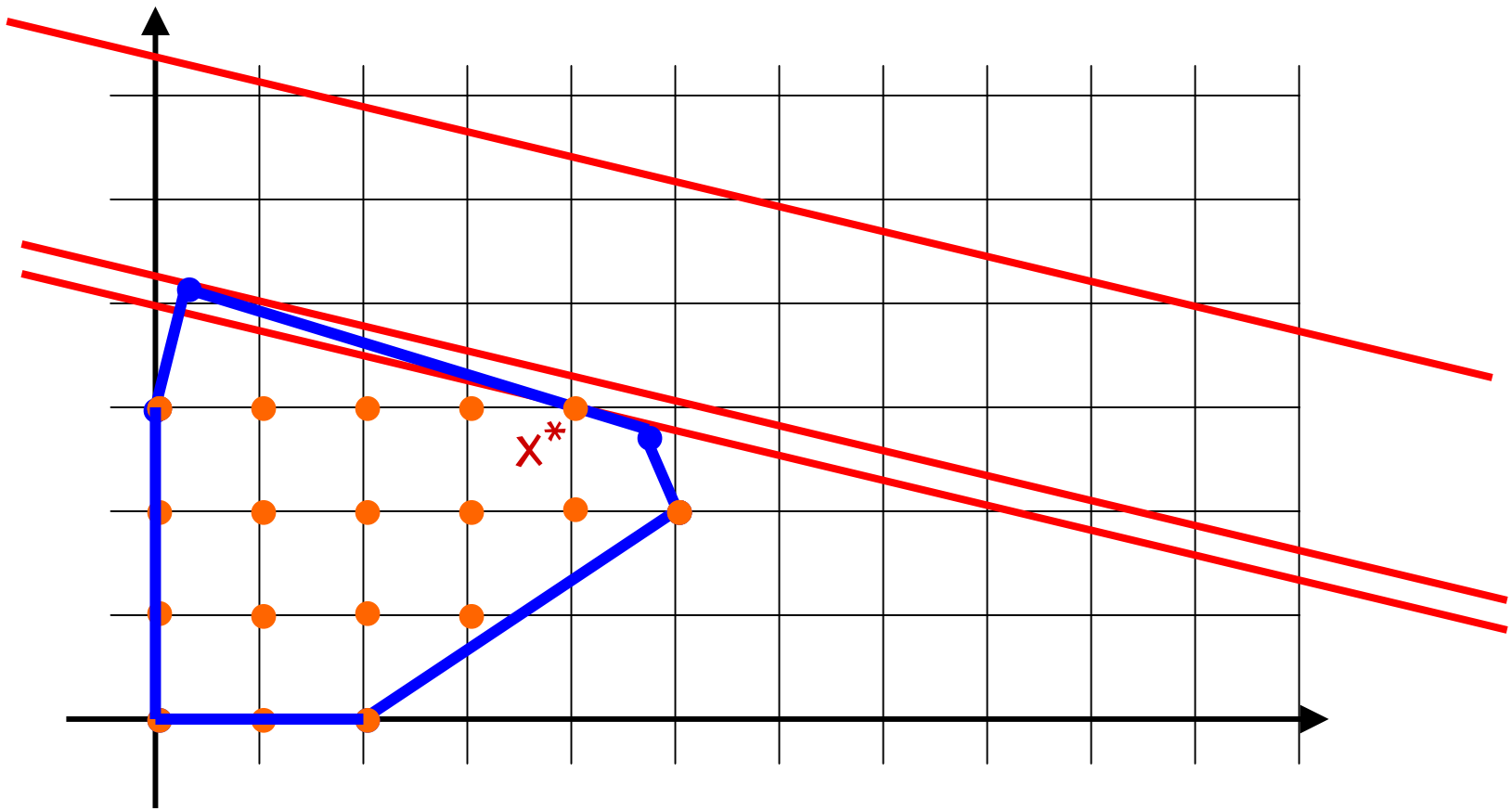
Caso 3:



O arredondamento nem sempre faz sentido, a solução obtida por arredondamento pode nem ser viável.

Solução gráfica

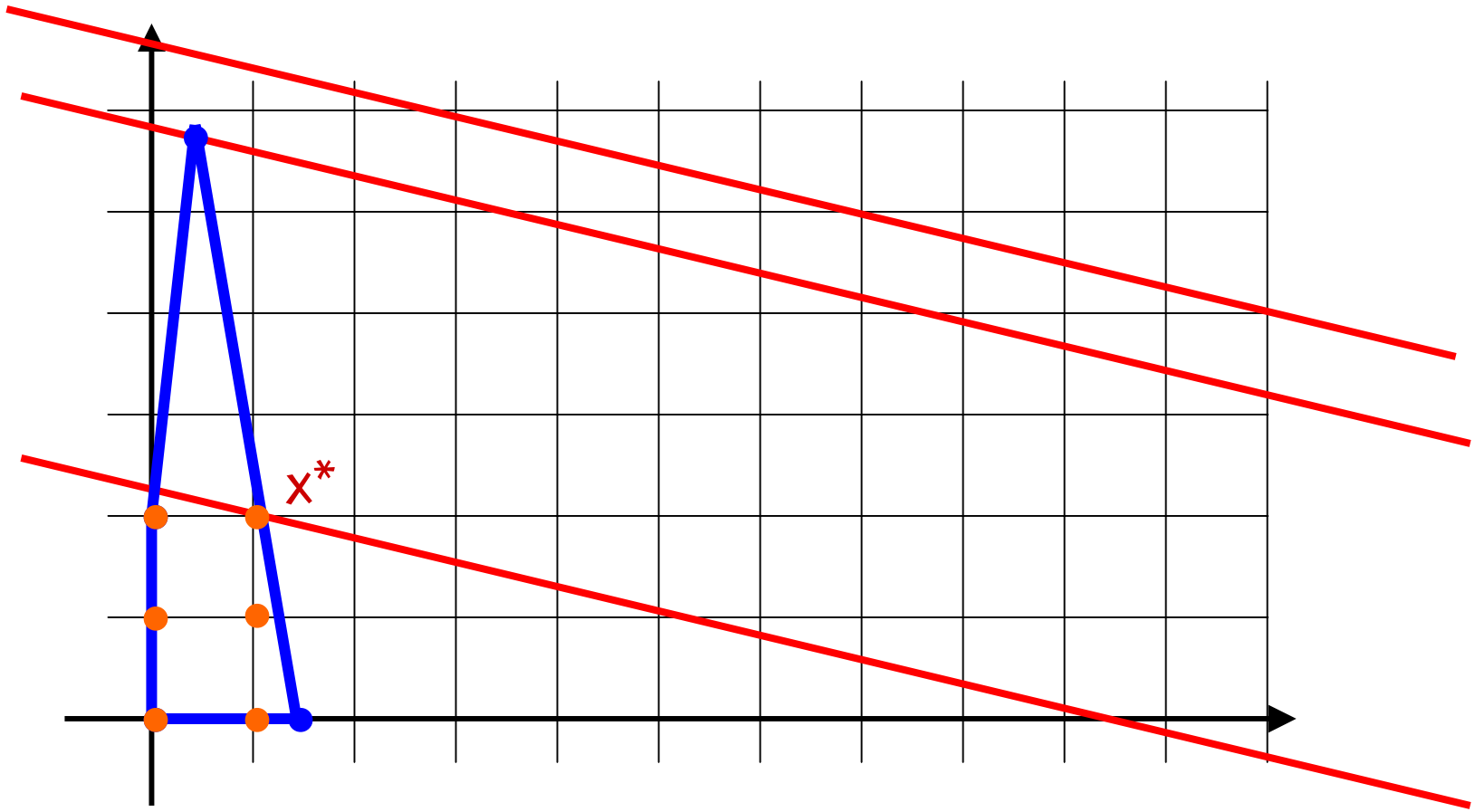
Caso 4:



Solução ótima contínua está muito afastada da solução ótima inteira.

Solução gráfica

Caso 5:



O valor do ótimo contínuo é muito diferente do valor do ótimo inteiro.

Solução gráfica

Caso 6:

O problema linear admite solução viável, mas o inteiro não.

Formulação de problemas

- (I) Problema de corte linear (revisitado): Uma fábrica necessita cortar uma fita de aço de 12 cm de largura em tiras de 2,4 cm, 3,4 cm e 4,5 cm de largura. As necessidades globais de tiras de cada comprimento são as seguintes:

Tipo	Largura (cm)	Demanda (m)
1	2,4	2500
2	3,4	4500
3	4,5	8000

Formulação de problemas

Formule um modelo que permita otimizar o consumo da fita a ser cortada, minimizando a perda de material.

Padrão	Tiras 1	Tiras 2	Tiras 3	Perda (cm)
1	5	0	0	0
2	3	1	0	1,4
3	3	0	1	0,3
4	2	2	0	0,4
5	1	0	2	0,6
6	0	3	0	1,8
7	0	2	1	0,7

Formulação de problemas

Observe entretanto que, nesta nova versão do problema, a fita não tem comprimento infinito, estando disponível em bobinas de comprimento igual a 400 metros. Cada bobina deve ser cortada segundo um único padrão de corte. Deseja-se minimizar o número de bobinas cortadas.

Formulação de problemas

Comprimento cortado de tiras de cada tipo segundo cada padrão:

Padrão	Tiras 1	Tiras 2	Tiras 3
1	2000	0	0
2	1200	400	0
3	1200	0	400
4	800	800	0
5	400	0	800
6	0	1200	0
7	0	800	400

Formulação de problemas

- Variáveis de decisão:

  x_i: número de bobinas cortadas pelo padrão i=1,...,7

- Restrições de demanda por tipo de fita:

  $2000\,x_1 + 1200\,x_2 + 1200\,x_3 + 800\,x_4 + 400\,x_5 \geq 2500$ tipo 1

  $400\,x_2 + 800\,x_4 + 1200\,x_6 + 800\,x_7 \geq 4500$ tipo 2

  $400\,x_3 + 800\,x_5 + 400\,x_7 \geq 8000$ tipo 3

- Função objetivo: minimizar $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$

Formulação de problemas

- (II) Problema de localização de facilidades: São dadas n potenciais localizações para a instalação de facilidades e m clientes que devem ser atendidos por estas facilidades. O custo fixo de instalar uma facilidade no local j é igual a c_j. Existe ainda um custo d_{ij} para atender um cliente i a partir da facilidade j. Cada cliente deve ser alocado a uma única facilidade. Determinar os locais onde devem ser instaladas as facilidades, de modo a minimizar o custo total.
- Situação comum, onde duas decisões são dependentes: um cliente só pode ser alocado a uma facilidade se ela tiver sido instalada.

Formulação de problemas

- Variáveis de decisão:

$$y_j = \begin{cases} 1, & \text{se é instalada uma facilidade no local j} \\ 0, & \text{caso contrário} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{se o cliente i é atendido pela facilidade j} \\ 0, & \text{caso contrário} \end{cases}$$

- Restrições:
  - Todo cliente tem que ser atendido por uma única facilidade.
  - Um cliente só pode ser atendido por uma facilidade instalada.

Formulação de problemas

minimizar $\sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij}$

sujeito a :

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i = 1,\ldots,m$$

$$x_{ij} \leq y_j, \qquad \forall i = 1,\ldots,m; \forall j = 1,\ldots,n$$

$$x_{ij}, y_j \in \{0,1\}, \qquad \forall i = 1,\ldots,m; \forall j = 1,\ldots,n$$

Formulação de problemas

- (III) Problema de localização capacitado (aplicação típica em localização de serviços urbanos): São dadas n potenciais localizações para a instalação de facilidades e m clientes que devem ser atendidos por estas facilidades. O custo fixo de instalar uma facilidade no local j é igual a c_j. Existe ainda um custo d_{ij} para atender um cliente i a partir da facilidade j. Cada cliente i tem uma demanda f_i e deve ser alocado a uma única facilidade, conhecendo-se ainda a capacidade e_j de atendimento de cada facilidade j. Determinar os locais onde devem ser instaladas as facilidades, de modo a minimizar o custo total.

Formulação de problemas

- Variáveis de decisão:

$$y_j = \begin{cases} 1, & \text{se é instalada uma facilidade no local j} \\ 0, & \text{caso contrário} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{se o cliente i é atendido pela facilidade j} \\ 0, & \text{caso contrário} \end{cases}$$

- Restrições:
  - Todo cliente tem que ser atendido por uma única facilidade.
  - Um cliente só pode ser atendido por uma facilidade instalada.
  - Deve ser respeitada a capacidade máxima de cada facilidade.

Formulação de problemas

minimizar $\sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij}$

sujeito a :

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i = 1,\ldots,m$$

$$x_{ij} \le y_j, \qquad \forall i = 1,\ldots,m; \forall j = 1,\ldots,n$$

$$\sum_{i=1}^{m} f_i x_{ij} \le e_j y_j, \forall j = 1,\ldots,n$$

$$x_{ij}, y_j \in \{0,1\}, \qquad \forall i = 1,\ldots,m; \forall j = 1,\ldots,n$$

Formulação de problemas

- (IV) Transformação de variáveis: Como modelar uma situação onde uma variável x só pode assumir valores dentro de um conjunto de valores $\{a_1, a_2, \ldots, a_m\}$?
- Definir as variáveis binárias $y_j \in \{0,1\}$, j=1, ...,m.
- Fazer uma substituição de variáveis e criar novas restrições:

$$x = \sum_{j=1}^{m} a_j y_j$$

$$\sum_{j=1}^{m} y_j = 1$$

Formulação de problemas

- (V) Transformação de variáveis: Como modelar com variáveis binárias 0-1 uma situação onde uma variável x pode assumir qualquer valor inteiro?
- Supõe-se que a variável seja limitada, por exemplo $x \leq A$ com $A = 2^k-1$.
- Definir as variáveis binárias $y_j \in \{0,1\}$, j=1,...,k.
- Fazer uma substituição de variáveis:

$$x = \sum_{j=1}^{k} 2^{j-1} y_j$$

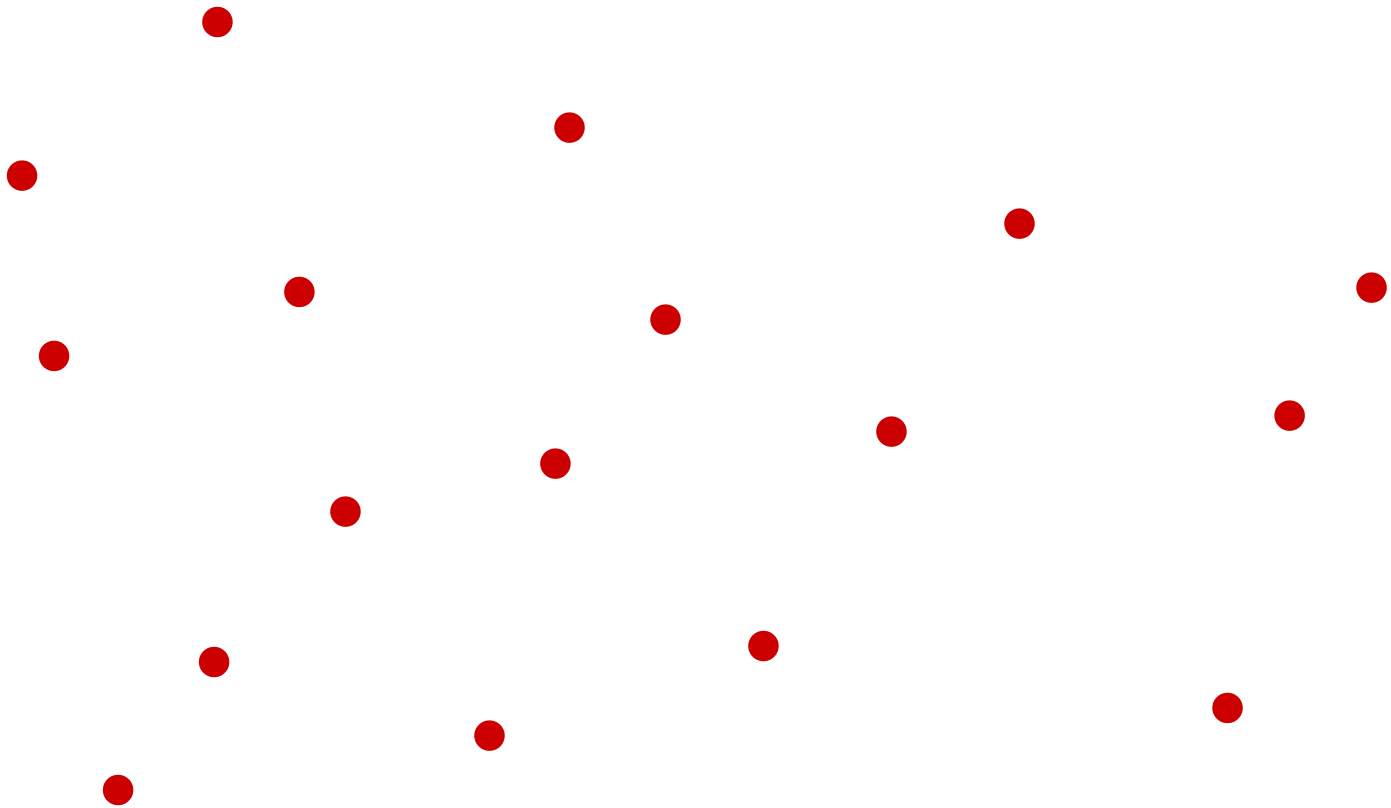
Formulação de problemas

- Sempre é possível transformar um problema de programação inteira com variáveis inteiras limitadas superiormente em um problema apenas com variáveis binárias 0-1.

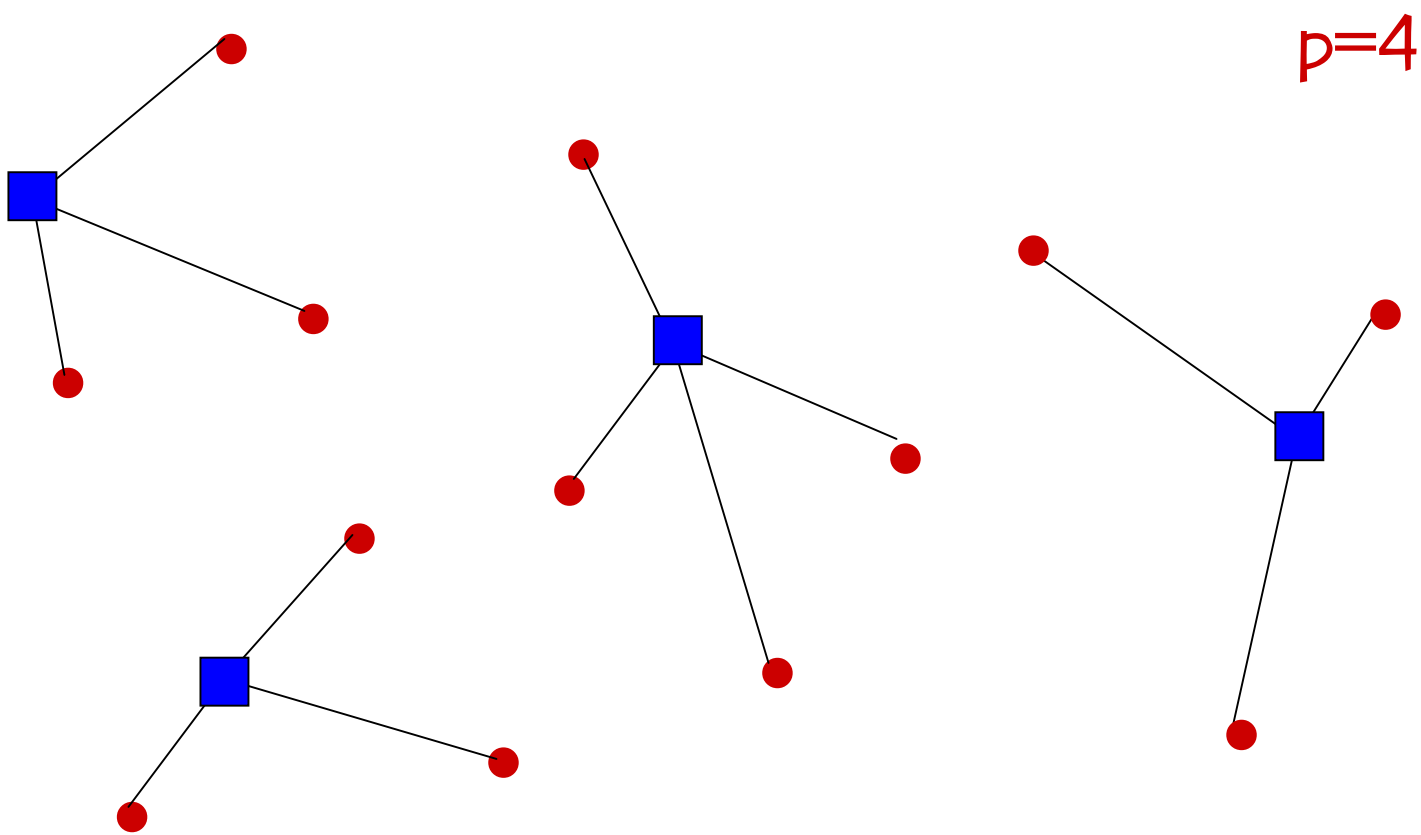
Formulação de problemas

- (VI) Problema das p-medianas (aplicações típicas em estatística, em problemas de classificação e agrupamento de dados): Seja uma amostragem formada por n objetos. Para dada par de objetos i e j, seja d_{ij} uma medida de sua dissimilaridade. Agrupar os objetos em p classes representativas, de modo que a soma das dissimilaridades de cada objeto ao representante de sua classe seja minimizada.

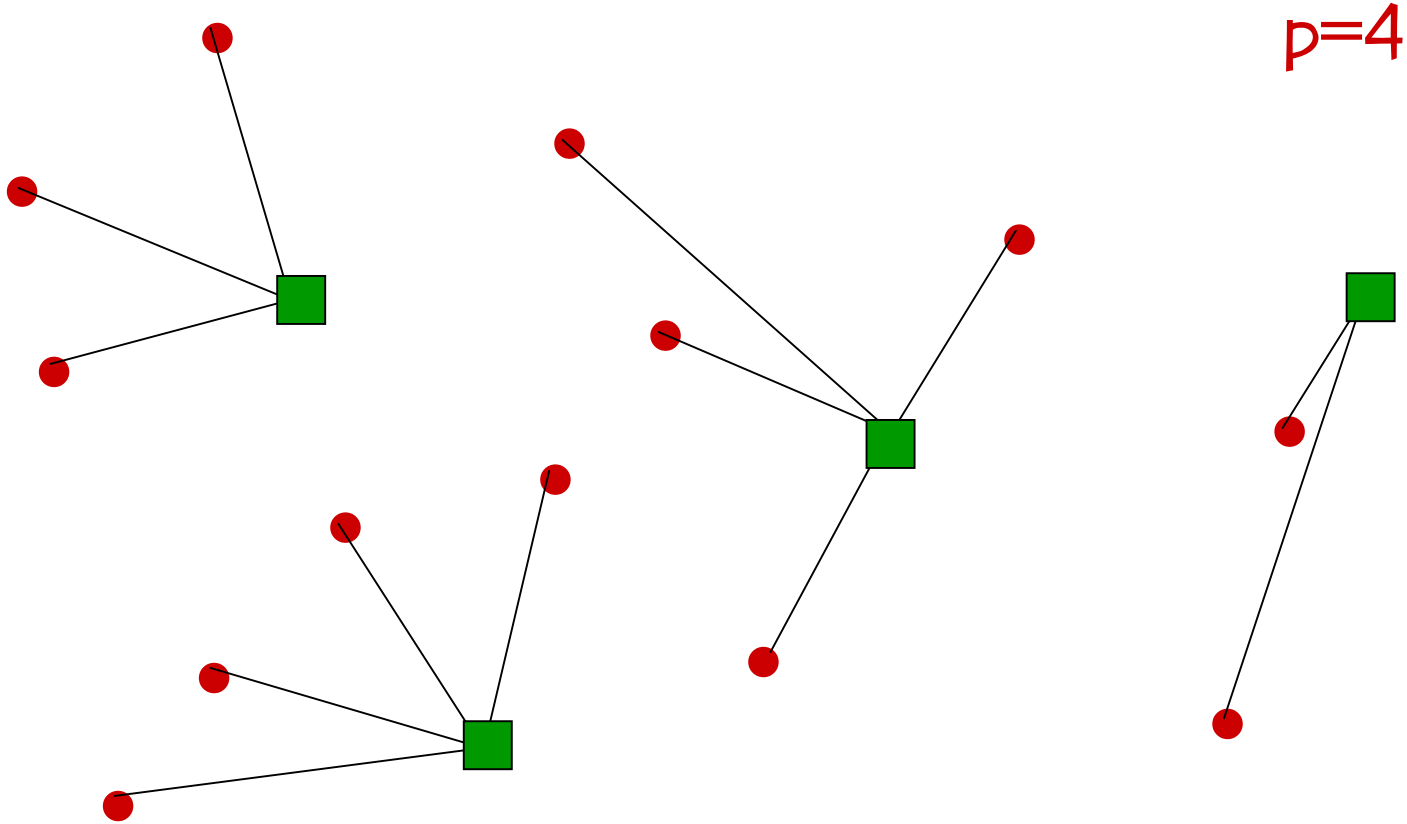
Formulação de problemas



Formulação de problemas



Formulação de problemas



Formulação de problemas

- Variáveis de decisão:

$$y_j = \begin{cases} 1, & \text{se o objeto j é escolhido como representante} \\ 0, & \text{caso contrário} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{se o objeto i é associado à classe representada pelo objeto j} \\ 0, & \text{caso contrário} \end{cases}$$

- Restrições:
  - Todo objeto está associado a um único representante.
  - Um objeto só pode estar associado a um objeto selecionado.
  - Há p objetos escolhidos como representantes.

Formulação de problemas

minimizar $\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}x_{ij}$

sujeito a :

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i = 1,\ldots,m$$

$$x_{ij} \le y_j, \qquad \forall i = 1,\ldots,m; \forall j = 1,\ldots,n$$

$$\sum_{j=1}^{n} y_j = p$$

$$x_{ij}, y_j \in \{0,1\}, \qquad \forall i = 1,\ldots,m; \forall j = 1,\ldots,n$$

Formulação de problemas

- (VII) Problema da clique de peso máximo (aplicações em data mining, tarifação telefônica): Dado um grafo $G = [V,A]$ onde $V = \{1,...,n\}$ é o conjunto de vértices e A é o conjunto de arestas, diz-se que o subconjunto de nós $X \subseteq V$ é uma clique se para qualquer nós $i \in X$, $j \in X$, então a aresta $(i,j) \in A$. Atribui-se a cada nó $i \in V$ um peso w_i. Descrever um modelo para determinar os nós de uma clique de peso máximo.
- Variáveis de decisão:

  $x_i = 1$ se e somente se o nó i é selecionado

Formulação de problemas

- Seja um par de nós (i,j) que não está conectado por uma aresta.
- Então, apenas um destes nós poderá ser selecionado para pertencer a uma clique: $x_i + x_j \leq 1 \quad \forall (i,j) \notin A$

$$\text{maximizar} \sum_{i=1}^{n} w_i . x_i$$

$$\text{sujeito a} : x_i + x_j \leq 1, \quad \forall (i,j) \notin A$$

$$x_i \in \{0,1\}, \quad \forall i \in V$$

Formulação de problemas

- (VIII) Problema do conjunto independente de peso máximo (aplicações em data mining, construção de portfolios): Dado um grafo $G = [V,A]$ onde $V = \{1,...,n\}$ é o conjunto de vértices e A é o conjunto de arestas, diz-se que o subconjunto de nós $X \subseteq V$ é um conjunto independente se para qualquer nós $i \in X$, $j \in X$, então a aresta $(i,j) \notin A$. Atribui-se a cada nó $i \in V$ um peso w_i. Descrever um modelo para determinar os nós de um conjunto independente de peso máximo.
- Variáveis de decisão:

  $x_i = 1$ se e somente se o nó i é selecionado

Formulação de problemas

- Seja um par de nós (i,j) conectado por uma aresta.
- Então, apenas um destes nós poderá ser selecionado para pertencer a um conjunto estável: $x_i + x_j \leq 1 \quad \forall (i,j) \in A$

$$\text{maximizar} \sum_{i=1}^{n} w_i . x_i$$

$$\text{sujeito a} : x_i + x_j \leq 1, \quad \forall (i,j) \in A$$

$$x_i \in \{0,1\}, \quad \forall i \in V$$

Formulação de problemas

- (IX) Problemas de recobrimento, particionamento e empacotamento de conjuntos: Sejam M = {1, ..., m} e N = {1, ..., n}. Seja ainda M_1, M_2, ..., M_n uma coleção de subconjuntos de M. Associa-se a cada subconjunto M_j um custo c_j, j=1,...,n. Seja F $\subseteq$ N. Então:

F é uma cobertura de M se $\bigcup_{j \in F} M_j = M$.

F é um empacotamento de M se $M_j \bigcap M_k = \{\}, \forall j,k \in F, j \neq k$.

F é uma partição de M se for uma cobertura e um empacotamento de M.

Formulação de problemas

- Peso do subconjunto F $\subseteq$ N é definido como $\sum_{j \in F} c_j$
- Problema de:
  - recobrimento: encontrar uma cobertura de peso mínimo
  - empacotamento: encontrar um empacotamento de peso máximo
  - particionamento: encontrar uma partição de peso mínimo (ou máximo)
- Formulação: definir os coeficientes $a_{ij} = \begin{cases} 1, & \text{se } i \in M_j \\ 0, & \text{caso contrário} \end{cases}$

Formulação de problemas

- Problema de recobrimento:

$$\text{minimizar} \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeito a}: \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad \forall i = 1,\ldots,m$$

$$x_j \in \{0,1\}, \quad \forall i = 1,\ldots,n$$

- Problema de empacotamento:

$$\text{maximizar} \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeito a}: \sum_{j=1}^{n} a_{ij} x_j \leq 1, \quad \forall i = 1,\ldots,m$$

$$x_j \in \{0,1\}, \quad \forall i = 1,\ldots,n$$

Formulação de problemas

- Problema de particionamento:

$$\text{minimizar} \sum_{j=1}^{n} c_j x_j$$

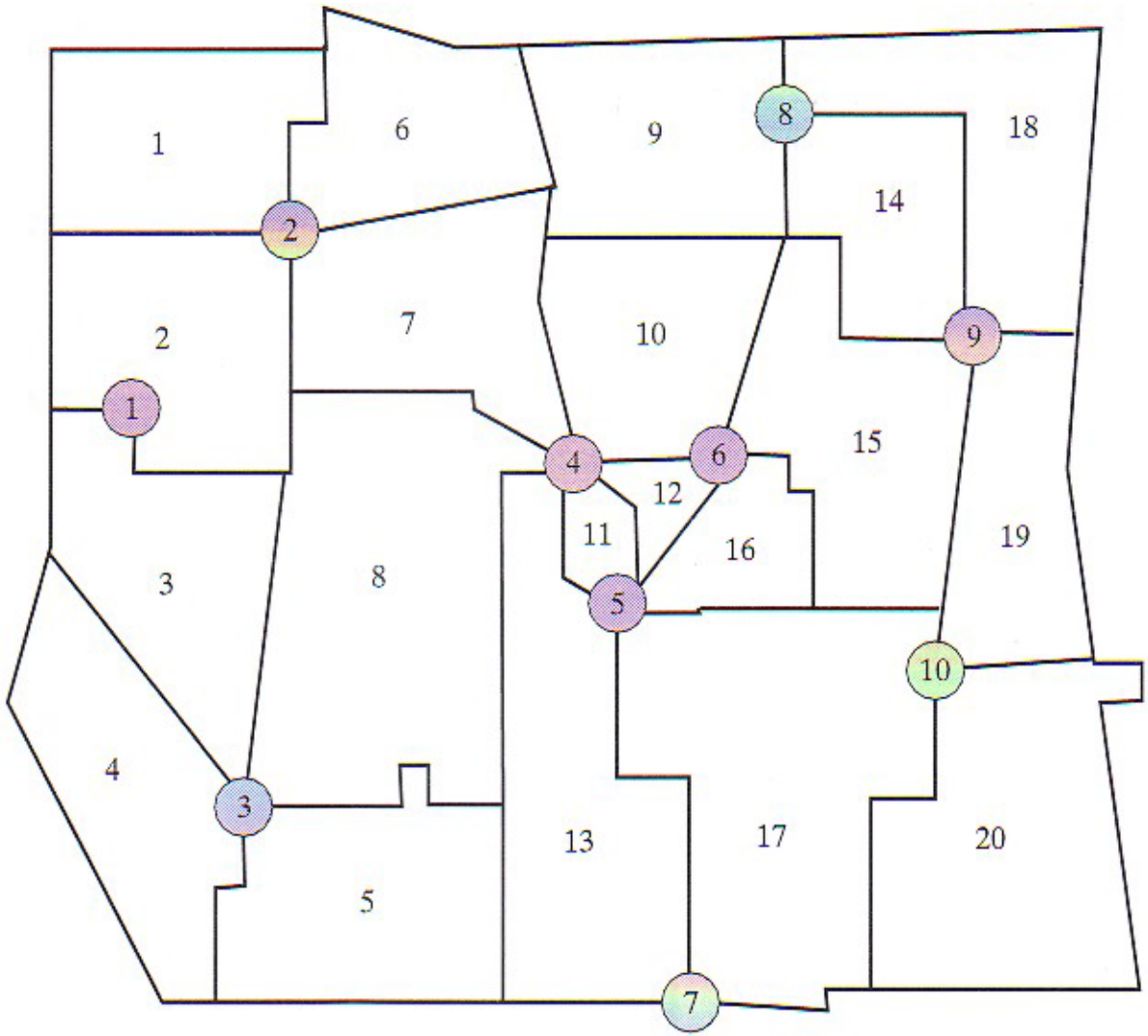
$$\text{sujeito a}: \sum_{j=1}^{n} a_{ij} x_j = 1, \quad \forall i = 1,\ldots,m$$

$$x_j \in \{0,1\}, \quad \forall i = 1,\ldots,n$$

Formulação de problemas

- (X) Localização de serviços por recobrimento: Uma empresa de atendimento médico deseja determinar os locais onde deve instalar veículos para atendimento médico em uma cidade. Esta cidade está dividida em 20 distritos. Veículos podem ser instalados em 10 locais previamente selecionados, de modo que possam responder rapidamente a chamadas provenientes de qualquer distrito. Um veículo pode atender a chamadas de qualquer distrito adjacente à sua localização. Deseja-se minimizar o número de veículos a serem adquiridos para instalação nesses locais.

Formulação de problemas



Formulação de problemas

- Variáveis de decisão:

  $x_j = 1$ se e somente se um veículo é instalado no local j

Formulação de problemas

minimizar $\sum_{j=1}^{10} x_j$

sujeito a:

$x_2 \geq 1$

$x_1 + x_2 \geq 1$

$x_1 + x_3 \geq 1$

$x_3 \geq 1$

$x_3 \geq 1$

$x_2 \geq 1$

$x_2 + x_4 \geq 1$

$x_3 + x_4 \geq 1$

$x_8 \geq 1$

$x_4 + x_6 \geq 1$

$x_4 + x_5 \geq 1$

$x_4 + x_5 + x_6 \geq 1$

$x_4 + x_5 + x_7 \geq 1$

$x_8 + x_9 \geq 1$

$x_6 + x_9 \geq 1$

$x_5 + x_6 \geq 1$

$x_5 + x_7 + x_{10} \geq 1$

$x_8 + x_9 \geq 1$

$x_9 + x_{10} \geq 1$

$x_{10} \geq 1$

$x_1, \ldots, x_{10} \in \{0,1\}$

Formulação de problemas

- Problema de recobrimento: todas as restrições do tipo ≥ 1 e todos os coeficientes das restrições iguais a 0 ou 1
- Como construir uma solução viável?
  - Heurística:
    - Determinar a variável que cobre o maior número de linhas descobertas.
    - Fixá-la em um.
    - Eliminar as linhas cobertas do modelo.
    - Repetir os passos acima até que todas as linhas estejam cobertas.
    - $x_2 = x_3 = x_4 = x_6 = x_8 = x_{10} = 1$
- Solução ótima: $x_2 = x_3 = x_4 = x_6 = x_8 = x_{10} = 1$

Formulação de problemas

- Supõe-se agora que:
  - Há recursos para instalar veículos em no máximo quatro locais.
  - Para cada distrito j=1,...,20 existe uma medida da demanda ou da importância d_j de atendê-lo.
  - Deseja-se minimizar a importância dos distritos não atendidos.
- Variáveis de decisão:

  x_j = 1 se e somente se um veículo é instalado no local j

  y_i = 1 se e somente se o distrito i não está coberto

Formulação de problemas

minimizar $\sum_{i=1}^{20} d_i . y_i$

sujeito a:

$$x_2 + y_1 \geq 1$$
$$x_1 + x_2 + y_2 \geq 1$$
$$x_1 + x_3 + y_3 \geq 1$$
$$x_3 + y_4 \geq 1$$
$$x_3 + y_5 \geq 1$$
$$x_2 + y_6 \geq 1$$
$$x_2 + x_4 + y_7 \geq 1$$
$$x_3 + x_4 + y_8 \geq 1$$
$$x_8 + y_9 \geq 1$$
$$x_4 + x_6 + y_{10} \geq 1$$
$$x_4 + x_5 + y_{11} \geq 1$$
$$x_4 + x_5 + x_6 + y_{12} \geq 1$$
$$x_4 + x_5 + x_7 + y_{13} \geq 1$$
$$x_8 + x_9 + y_{14} \geq 1$$
$$x_6 + x_9 + y_{15} \geq 1$$
$$x_5 + x_6 + y_{16} \geq 1$$
$$x_5 + x_7 + x_{10} + y_{17} \geq 1$$
$$x_8 + x_9 + y_{18} \geq 1$$
$$x_9 + x_{10} + y_{19} \geq 1$$
$$x_{10} + y_{20} \geq 1$$
$$\sum_{j=1}^{10} x_j \leq 4$$
$$x_1, \ldots, x_{10}, y_1, \ldots, y_{20} \in \{0,1\}$$

Formulação de problemas

- (XI) Problema de alocação de tripulações: Uma empresa aérea deseja alocar tripulações a um conjunto de vôos, representados como no diagrama a seguir. Cada rotação é uma seqüência de vôos cobertos pela mesma tripulação durante um certo período de tempo estipulado por leis trabalhistas e acordos sindicais, originando-se e terminando na cidade onde esta tripulação está baseada. A lista de rotações possíveis, estabelecida por outro modelo baseado em métodos enumerativos, também é fornecida como dado de entrada. O cálculo do custo de cada

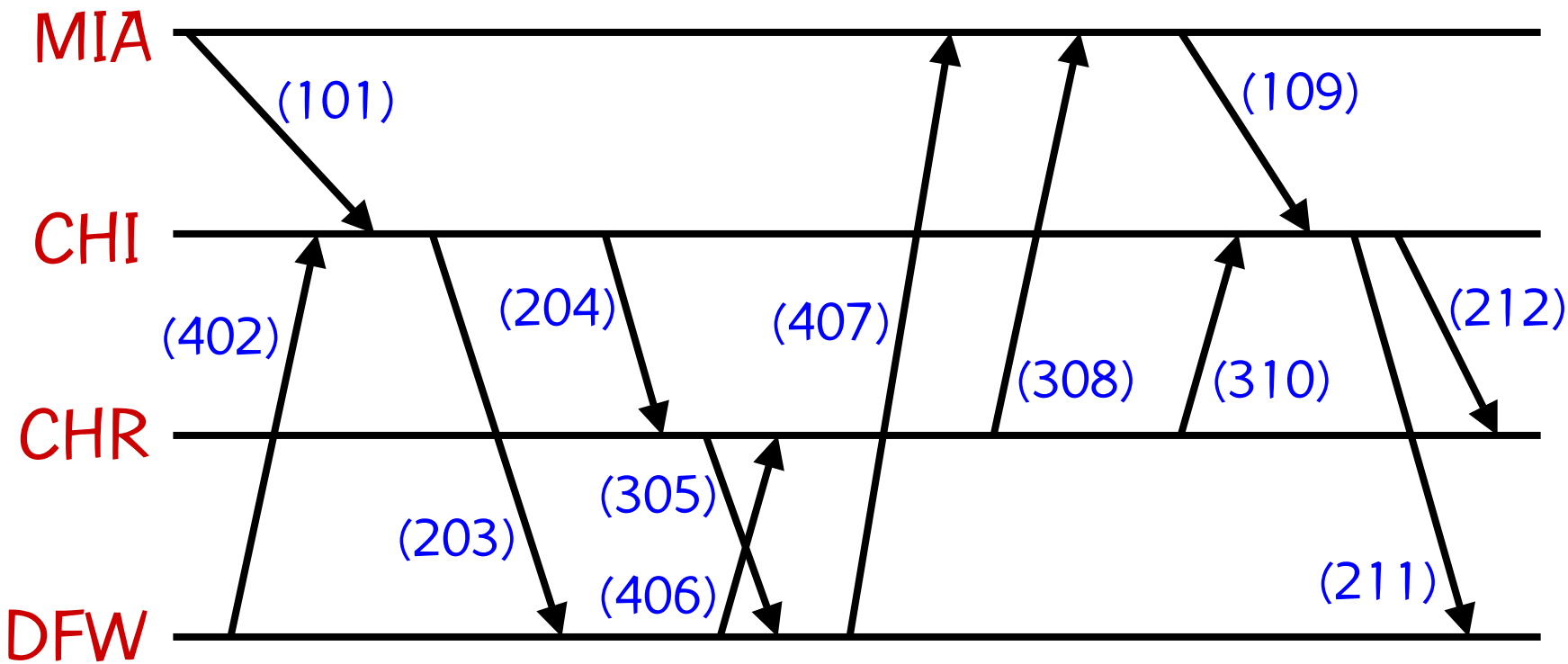
Formulação de problemas

rotação também é complexo, envolvendo o salário das tripulações, horas-extras, despesas de hotel e transporte, etc. Deseja-se atribuir tripulações aos vôos, de modo a minimizar os custos com as tripulações.

- Área extremamente fértil em aplicações de programação inteira: alocação de tripulações, alocação de aeronaves, alocação de portões de embarque, etc.
- Variáveis de decisão:

  $x_j = 1$ se e somente se a rotação j é escolhida

Formulação de problemas



Formulação de problemas

j	vôos	custo	j	vôos	custo
1	101-203-406-308	2900	9	305-407-109-212	2600
2	101-203-407	2700	10	308-109-212	2050
3	101-204-305-407	2600	11	402-204-305	2400
4	101-204-308	3000	12	402-204-310-211	3600
5	203-406-310	2600	13	406-308-109-211	2550
6	203-407-109	3150	14	406-310-211	2650
7	204-305-407-109	2550	15	407-109-211	2350
8	204-308-109	2500	-	-	-

Formulação de problemas

minimizar $2900x_1 + 2700x_2 + 2600x_3 + 3000x_4 + 2600x_5 +$
$+ 3150x_6 + 2550x_7 + 2500x_8 + 2600x_9 + 2050x_{10} + 2400x_{11} +$
$+ 3600x_{12} + 2550x_{13} + 2650x_{14} + 2350x_{15}$

sujeito a :

$x_1 + x_2 + x_3 + x_4 = 1$

$x_6 + x_7 + x_8 + x_9 + x_{10} + x_{13} + x_{15} = 1$

$x_1 + x_2 + x_5 + x_6 = 1$

$x_3 + x_4 + x_7 + x_8 + x_{11} + x_{12} = 1$

$x_{12} + x_{13} + x_{14} + x_{15} = 1$

$x_9 + x_{10} = 1$

$x_3 + x_7 + x_9 + x_{11} = 1$

$x_1 + x_4 + x_8 + x_{10} + x_{13} = 1$

$x_5 + x_{12} + x_{14} = 1$

$x_{11} + x_{12} = 1$

$x_1 + x_5 + x_{13} + x_{14} = 1$

$x_2 + x_3 + x_6 + x_7 + x_9 + x_{15} = 1$

$x_1, \ldots, x_{15} \in \{0,1\}$

Formulação de problemas

- Problema de particionamento: todas as restrições do tipo = 1 e todos os coeficientes das restrições iguais a 0 ou 1
- Como construir uma solução viável?
  - Heurística:
    - Fixar uma variável em um.
    - Fixar em zero todas as demais variáveis que aparecem nas mesmas linhas que a anterior.
    - Repetir os passos acima até que todas as variáveis estejam fixadas.
    - Pode não dar certo: por exemplo, fixar $x_1 = 1$ e em seguida $x_7 = 1 \Rightarrow x_9 = x_{10} = 0 \Rightarrow$ a sexta igualdade não pode ser satisfeita.
- Solução ótima: $x_1 = x_9 = x_{12} = 1$

Formulação de problemas

- (XII) Problema do caixeiro viajante: Considere um conjunto de n cidades N = {1, 2, ..., n}. Seja c_{ij} a distância entre a cidade i e a cidade j. Um vendedor encontra-se inicialmente em uma cidade qualquer. Ele deve visitar todas as cidades uma e exatamente uma única vez, efetuar uma operação comercial em cada uma delas e retornar à cidade inicial, percorrendo a menor distância possível. Formular como um problema de programação inteira.
- Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{se a cidade } j \text{ é visitada imediatamente após a cidade } i \\ 0, & \text{caso contrário} \end{cases}$$

Formulação de problemas

- Função objetivo:

$$\text{minimizar} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

- Cada cidade deve ser precedida por outra:

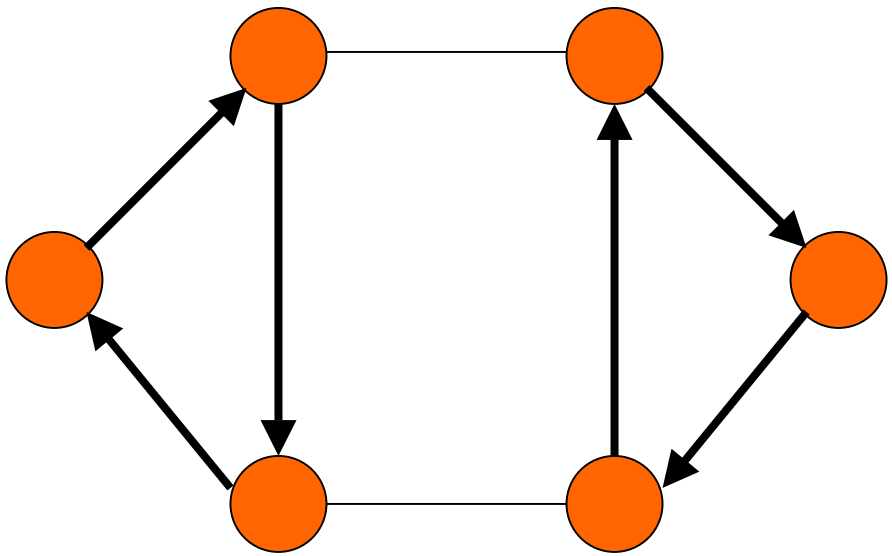
$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j = 1,\ldots,n$$

- Cada cidade deve ser sucedida por outra:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i = 1,\ldots,n$$

Formulação de problemas

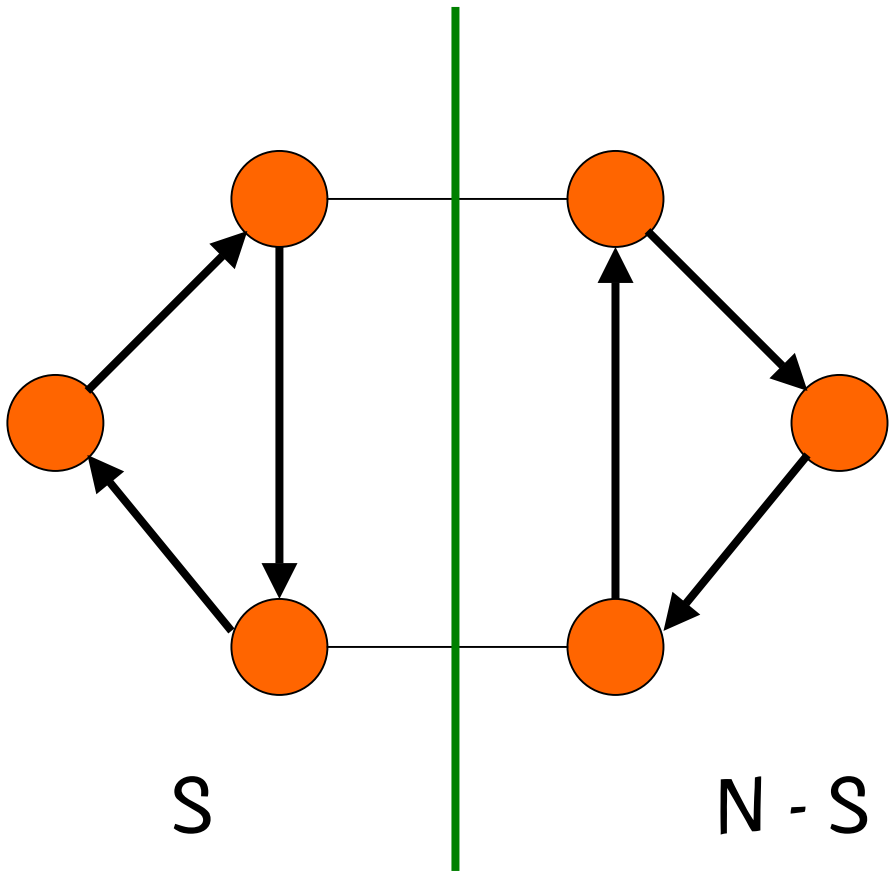
- As restrições anteriores garantem que um único ciclo será formado e que todas as cidades serão visitadas?



Não!

Formulação de problemas

- Cada cidade tem que ser atingida:



Para que as cidades do lado direito sejam atingidas, é necessário que pelo menos uma variável correspondente a um deslocamento da esquerda para a direita seja igual a um.

Formulação de problemas

minimizar $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$

sujeito a :

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j = 1,\ldots,n$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i = 1,\ldots,n$$

restrições de eliminação de "subtours" ➡

$$\sum_{i\in S}\sum_{j\in N-S} x_{ij} \geq 1, \qquad \forall S \subset N, S \neq \{\}, S \neq N$$

$$x_{ij} \in \{0,1\}, \qquad \forall i = 1,\ldots,n;\ j = 1,\ldots,n$$

Problema: número elevado (exponencial) de restrições!

Formulação de problemas

- As restrições de eliminação de "subtours" não precisam ser todas incluídas no modelo de uma única vez, mas sim apenas a medida em que forem necessárias para eliminar uma solução que viole uma delas.

Formulação de problemas

- Aplicação: Uma máquina deve realizar n tarefas com tempos de execução conhecidos. Seja c_{ij} o tempo de preparação (pouso de aviões, pinturas em oficinas) da tarefa j caso ela seja executada imediatamente após a tarefa i. Determinar a ordem em que as tarefas devem ser executadas de modo a minimizar o tempo total de execução (constante) e de preparação (dependente da ordem) corresponde a resolver um problema do caixeiro viajante).

Formulação de problemas

- (XIII) Problema de escalonamento (seqüenciamento) com tempos de preparação ("setup"): Uma máquina pode executar m operações, numeradas de 1 a m. Cada operação j exige uma única ferramenta j. A máquina pode guardar B ferramentas em sua caixa de ferramentas, com B < m. Montar ou desmontar a ferramenta j na caixa de ferramentas exige s_j unidades de tempo de preparação. A cada instante, apenas uma ferramenta pode ser montada ou desmontada. No início do dia, n tarefas aguardam processamenro nesta máquina. Cada tarefa i exige

Formulação de problemas

múltiplas operações. Seja J_i o conjunto de operações exigidas pela tarefa i e assuma, como hipótese simplificadora, que para cada tarefa i o número de operações $|J_i|$ não é maior do que a capacidade B da caixa de ferramentas da máquina. Antes da máquina começar o processamento da tarefa i, todas as ferramentas pertencentes ao conjunto J_i precisam ser montadas na máquina. Se uma ferramenta $j \in J_i$ já está carregada na máquina, evita-se seu tempo de preparação. Se a ferramenta $j \in J_i$ ainda não está carregada, então é

Formulação de problemas

necessário carregá-la, possivelmente (caso a caixa de ferramentas esteja cheia neste instante) depois de desinstalar alguma ferramenta que a tarefa i não exija. Uma vez que todas as ferramentas estejam preparadas, então as $|J_i|$ operações da tarefa i são processadas. Observe que, em razão de diversas tarefas exigirem ferramentas em comum e devido à capacidade máxima da caixa de ferramentas, os tempos de preparação antes de cada tarefa dependem da ordem em que estas são executadas. Deseja-se formular um modelo de programação inteira

Formulação de problemas

para determinar a ordem ótima de execução das tarefas, minimizando o tempo total de preparação para completar todas as n tarefas. Assume-se ainda que no início do dia a caixa de ferramentas está completamente vazia.

- Variáveis de decisão:

$$x_{ir} = \begin{cases} 1, & \text{se a tarefa } i \text{ é a } r \text{ - ésima a ser processada} \\ 0, & \text{caso contrário} \end{cases}$$

$$y_{jr} = \begin{cases} 1, & \text{se a ferramenta } j \text{ está na caixa de ferramentas} \\ & \text{quando a } r \text{ - ésima tarefa é processada} \\ 0, & \text{caso contrário} \end{cases}$$

Formulação de problemas

- A caixa de ferramentas está vazia no início do dia:

$$y_{j0} = 0, \qquad \forall j = 1,\ldots,n$$

- Todas as tarefas devem ser processadas:

$$\sum_{r=1}^{n} x_{ir} = 1, \qquad \forall i = 1,\ldots,n$$

- A cada instante uma única tarefa pode ser processada:

$$\sum_{i=1}^{n} x_{ir} = 1, \qquad \forall r = 1,\ldots,n$$

Formulação de problemas

- Para que a tarefa i seja processada, todas as ferramentas no conjunto J_i devem estar instaladas na caixa:

$$x_{ir} \leq y_{jr}, \quad \forall r = 1,\ldots,n; \forall i = 1,\ldots,n; \forall j \in J_i$$

- Capacidade máxima da caixa de ferramentas:

$$\sum_{j=1}^{m} y_{jr} \leq B, \quad \forall r = 1,\ldots,n$$

Formulação de problemas

- Incorre-se em custos de preparação sempre que uma ferramenta é carregada ou descarregada:

$$\text{minimizar} \sum_{j=1}^{m} \sum_{r=1}^{n} s_j . \left| y_{jr} - y_{j,r-1} \right|$$

- Definir uma nova variável inteira z_{jr} e substituí-la na função objetivo:

$$\text{minimizar} \sum_{j=1}^{m} \sum_{r=1}^{n} s_j . z_{jr}$$

Formulação de problemas

- Para isto, é necessário adicionar novas restrições:

$$z_{jr} \geq y_{jr} - y_{j,r-1}, \quad \forall j = 1,\ldots,m; \forall r = 1,\ldots,n$$

$$z_{jr} \geq y_{j,r-1} - y_{jr}, \quad \forall j = 1,\ldots,m; \forall r = 1,\ldots,n$$

Formulação de problemas

minimizar $\sum_{j=1}^{m}\sum_{r=1}^{n} s_j . z_{jr}$

sujeito a : $x_{ir} \le y_{jr}, \quad \forall j \in J_i; \forall r = 1,...,n; \forall i = 1,...,n$

$$z_{jr} \ge y_{jr} - y_{j,r-1}, \qquad \forall j = 1,...,m; \forall r = 1,...,n$$

$$z_{jr} \ge y_{j,r-1} - y_{jr}, \qquad \forall j = 1,...,m; \forall r = 1,...,n$$

$$\sum_{r=1}^{n} x_{ir} = 1, \qquad \forall i = 1,...,n$$

$$\sum_{i=1}^{n} x_{ir} = 1, \qquad \forall r = 1,...,n$$

$$\sum_{j=1}^{m} y_{jr} \le B, \qquad \forall r = 1,...,n \qquad x_{ir}, y_{jr}, z_{jr} \in \{0,1\}$$

Formulação de problemas

- (XIV) Escalonamento de uma máquina com datas de liberação e prazos de execução: São conhecidas n tarefas que devem ser executadas em uma única máquina. Não se admite preempção, isto é, uma vez que uma tarefa começou a ser executada ela deve ser concluída sem interrupção nesta máquina. Apenas uma tarefa pode ser executada de cada vez. Para cada tarefa j=1,...,n, são conhecidas sua duração p_j, o instante (data) r_j em que a tarefa estará disponível para ser executada e o instante (data) d_j em que deverá estar pronta. Todas as datas são relativas ao momento presente (t = 0), algumas delas

Formulação de problemas

podendo ser negativas. Formular um modelo para determinar a ordem em que as tarefas devem ser executadas.

- Variáveis de decisão: como o escalonamento se dá ao longo do tempo, é natural usar como variáveis de decisão em problemas de escalonamento o instante de início ou de término de uma tarefa.

  x_j: instante de início da execução da tarefa j, em relação a t = 0 que é o instante inicial (agora).

Formulação de problemas

- Uma tarefa só pode ser executada depois que tiver sido liberada para processamento:

  $x_j \geq \max\{0, r_j\}$

- Sejam duas tarefas i e j quaisquer. Então, uma das duas restrições abaixo deve ser verificada, para que não existam conflitos de tempo de execução (uma deve terminar antes da outra começar):

  ou (início de i + duração de i $\leq$ início de j)

  ou (início de j + duração de j $\leq$ início de i)

  Como modelar este tipo de situação?

Formulação de problemas

- Variáveis de decisão:

  $y_{ij} = 1$ se e somente se a tarefa i começa antes da tarefa j

- Um par de restrições disjuntivas pode ser utilizado para representar que uma tarefa começa após a outra:

  $x_i + p_i \leq x_j + M.(1-y_{ij})$

  $x_j + p_j \leq x_i + M.y_{ij}$

  Para cada par de tarefas (i,j), só é necessário escrever este par de restrições para j > i (de modo a evitar repetições desnecessárias).

Formulação de problemas

- Término de cada tarefa:

  $x_j + p_j \leq d_j$

- Freqüentemente estas condições não são incluídas como restrições, mas sim penalizadas na função objetivo.
- Função objetivo?

Formulação de problemas

- Diversas funções objetivo são empregadas em problemas de escalonamento:
  - tempo máximo de término: minimizar $\max_{j=1,\ldots,n} \{x_j+p_j\}$
  - tempo médio de término: minimizar $(1/n).\sum_{j=1,\ldots,n} \{x_j+p_j\}$
  - tempo máximo de permanência: minimizar $\max_{j=1,\ldots,n} \{x_j+p_j-r_j\}$
  - tempo médio de permanência: minimizar $(1/n).\sum_{j=1,\ldots,n} \{x_j+p_j-r_j\}$
  - atraso máximo: minimizar $\max_{j=1,\ldots,n} \{x_j+p_j-d_j\}$
  - atraso médio: minimizar $(1/n).\sum_{j=1,\ldots,n} \{x_j+p_j-d_j\}$

Formulação de problemas

- Como se trata uma função objetivo do tipo abaixo?

  minimizar $\max_{j=1,\ldots,n} \{x_j+p_j\}$

- Transformar o problema em:

  minimizar z

  sujeito a:

  $$z \geq x_1 + p_1$$
  $$z \geq x_2 + p_2$$
  $$\ldots$$
  $$z \geq x_n + p_n$$

Formulação de problemas

- (XV) Problema de coloração de grafos (aplicação em problemas de seqüenciamento de tarefas): Dado um grafo $G = [V,A]$ onde $V = \{1,...,n\}$ é o conjunto de vértices e A é o conjunto de arestas, colorir os nós de G com um número mínimo de cores de modo que nós adjacentes sejam coloridos com cores diferentes.

Formulação de problemas

- Aplicação: Deseja-se marcar as datas dos exames das disciplinas em uma escola no menor número possível de dias, de modo que cada aluno faça uma única prova no mesmo dia. Disciplinas envolvidas em conflitos (isto é, cursadas por pelo menos um aluno em comum) devem ter suas provas marcadas em dias diferentes.
- Variáveis binárias de decisão:

  $y_j = 1$ se e somente se a cor j é utilizada

  $x_{ij} = 1$ se e somente se o nó i é colorido com a cor j

Formulação de problemas

- Minimizar o número de cores:

$$\text{minimizar} \sum_{j=1}^{n} y_j$$

- Um nó só pode ser colorido com uma cor se ela for usada:

$$x_{ij} \leq y_j, \quad \forall i = 1,\ldots,n; \forall j = 1,\ldots,n$$

- Um nó tem que ser colorido com alguma cor:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i = 1,\ldots,n$$

- Nós adjacentes devem ser coloridos com cores diferentes:

$$x_{ij} + x_{kj} \leq 1, \quad \forall j = 1,\ldots,n; \forall (i,k) \in A$$

Formulação de problemas

- (XVI) Um problema de decentralização: Uma cadeia de supermercados deseja decentralizar sua distribuição de provisões. A administração considera abrir dois centros de distribuição, um grande (CD_1) e um pequeno (CD_2), em duas cidades diferentes. As cidades A, B e C são candidatas para a instalação de um centro de distribuição. Com relação à situação atual, as economias passíveis de serem alcançadas (subsídios locais, por exemplo) com a instalação de um destes centros em uma destas cidades são dadas por:

Formulação de problemas

	A	B	C
CD grande	1000	1200	1500
CD pequeno	400	700	800

Há tráfego entre os dois CDs devido a serviços em comum (por exemplo, compartilham o mesmo serviço de manutenção). Os custos de transporte entre as três cidades devido a serviços em comum são dados pela tabela que se segue:

Formulação de problemas

	A	B	C
A	-	1100	1500
B	1100	-	1600
C	1500	1600	-

Decidir os melhores locais para a instalação dos dois centros de distribuição, considerando as economias e os custos de transporte.

Formulação de problemas

- Notação:
  - A = 1, B = 2, C = 3
  - E_{ij} = economias quando o CD_i está localizado na cidade j
  - T_{jl} = custo de transporte entre as cidades j e l
- Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{se o centro i é instalado na cidade j} \\ 0, & \text{caso contrário} \end{cases}$$

Formulação de problemas

- Cada centro de distribuição tem que ser instalado:

$$\sum_{j=1}^{3} x_{ij} = 1, \quad \forall i = 1,2$$

- Cada cidade recebe no máximo um centro de distribuição:

$$\sum_{i=1}^{2} x_{ij} \leq 1, \quad \forall j = 1,2,3$$

- Restrições de integralidade:

$$x_{ij} \in \{0,1\}, \quad \forall i = 1,2; \forall j = 1,2,3$$

Formulação de problemas

- Economias:

$$\sum_{i=1}^{2}\sum_{j=1}^{3} E_{ij}.x_{ij}$$

- Custo de transporte:

$$\sum_{i=1}^{2}\sum_{k=1}^{2}\sum_{j=1}^{3}\sum_{l=1}^{3} T_{jl}.x_{ij}.x_{kl}$$

- Função objetivo:

minimizar $\sum_{i=1}^{2}\sum_{j=1}^{3} E_{ij}.x_{ij} - \sum_{i=1}^{2}\sum_{k=1}^{2}\sum_{j=1}^{3}\sum_{l=1}^{3} T_{jl}.x_{ij}.x_{kl}$

Formulação de problemas

- Termo quadrático na função objetivo:

$$\text{minimizar } \sum_{i=1}^{2}\sum_{j=1}^{3} E_{ij}.x_{ij} - \sum_{i=1}^{2}\sum_{k=1}^{2}\sum_{j=1}^{3}\sum_{l=1}^{3} T_{jl}.x_{ij}.x_{kl}$$

- Linearização do termo quadrático através de variáveis binárias:

$$z_{ijkl} = \begin{cases} 1, & \text{se } x_{ij} = 1 \text{ e } x_{kl} = 1 \\ 0, & \text{caso contrário} \end{cases}$$

- O termo quadrático pode ser reescrito como:

$$x_{ij} + x_{kl} - 2.z_{ijkl} \geq 0$$

$$x_{ij} + x_{kl} - z_{ijkl} \leq 1.$$

Formulação de problemas

minimizar $\sum_{i=1}^{2}\sum_{j=1}^{3} E_{ij}.x_{ij} - \sum_{i=1}^{2}\sum_{k=1}^{2}\sum_{j=1}^{3}\sum_{l=1}^{3} T_{jl}.v_{ijkl}$

sujeito a :

$$\sum_{j=1}^{3} x_{ij} = 1, \qquad \forall i = 1,2$$

$$\sum_{i=1}^{2} x_{ij} \leq 1, \qquad \forall j = 1,2,3$$

$$x_{ij} + x_{kl} - 2.z_{ijkl} \geq 0, \qquad \forall i = 1,2; \forall k = 1,2; \forall j = 1,2,3; \forall l = 1,2,3$$

$$x_{ij} + x_{kl} - z_{ijkl} \leq 0, \qquad \forall i = 1,2; \forall k = 1,2; \forall j = 1,2,3; \forall l = 1,2,3$$

$$x_{ij}, z_{ijkl} \in \{0,1\}, \qquad \forall i = 1,2; \forall k = 1,2; \forall j = 1,2,3; \forall l = 1,2,3$$

Formulação de problemas

- A substituição de variáveis binárias freqüentemente permite linearizar e tratar de forma exata modelos quadráticos!

Exercício

- (XVII) Problema de planejamento da capacidade (revisitado): Um estado deseja fazer seu planejamento de capacidade instalada. A previsão de demanda de capacidade é igual a d_t MW para cada ano t=1,...,T. A capacidade existente em usinas a óleo e que estará disponível em cada ano t=1,...,T é igual a e_t. Há duas alternativas possíveis para a expansão de capacidade: usinas nucleares e usinas a carvão. Há um custo de capital igual a c_t por MW de usinas a carvão que tornam-se disponíveis no ano t. O custo de capital correspondente para usinas nucleares é n_t. A capacidade em usinas

Exercício

nucleares não pode ser superior a 20% da capacidade total instalada. Usinas a carvão têm vida útil de 20 anos, enquanto usinas nucleares têm vida útl de 15 anos. Deseja-se planejar a expansão de custo mínimo. Procurem criar e modelar uma versão mais realista deste problema, usando variáveis de decisão.

Formulação de problemas

- Boa formulação em PL: pequeno número de variáveis e de restrições
- Disponibilidade de bons algoritmos para PL: a escolha de uma formulação, embora importante, não afeta criticamente nossa capacidade de resolver um problema.
- Situação diferente em programação inteira!
- Conceito importante: relaxação linear de um problema de programação inteira

Formulação de problemas

- Modelo de programação linear inteira mista:

$$c.x^* + d.y^* = \text{mínimo } c.x + d.y$$

$$\text{sujeito a : } A.x + By = b$$

$$x, y \geq 0$$

$$x \text{ inteiro}$$

- Relaxação linear:

$$\bar{c}.\bar{x} + \bar{d}.\bar{y} = \text{mínimo } c.x + d.y$$

$$\text{sujeito a : } A.x + By = b$$

$$x, y \geq 0$$

Formulação de problemas

- Problema de localização de facilidades (revisitado):

minimizar $\sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij}$

sujeito a :

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i = 1,\ldots,m$$

$$x_{ij} \le y_j, \qquad \forall i = 1,\ldots,m; \forall j = 1,\ldots,n \qquad (1)$$

$$x_{ij}, y_j \in \{0,1\}, \qquad \forall i = 1,\ldots,m; \forall j = 1,\ldots,n$$

Formulação de problemas

- Formulação agregada alternativa:

minimizar $\sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij}$

sujeito a :

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i = 1,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} \le m.y_j, \quad \forall j = 1,\ldots,n \qquad (2)$$

$$x_{ij}, y_j \in \{0,1\}, \qquad \forall i = 1,\ldots,m; \forall j = 1,\ldots,n$$

Formulação de problemas

- Poliedros das soluções viáveis das relaxações lineares das duas formulações:

  - Com a restrição 1:

$$P_1 = \left\{(x,y) : \sum_{j=1}^{n} x_{ij} = 1, \forall i; x_{ij} \le y_j, \forall i,j; 0 \le x_{ij} \le 1,\ 0 \le y_j \le 1, \forall i,j\right\}$$

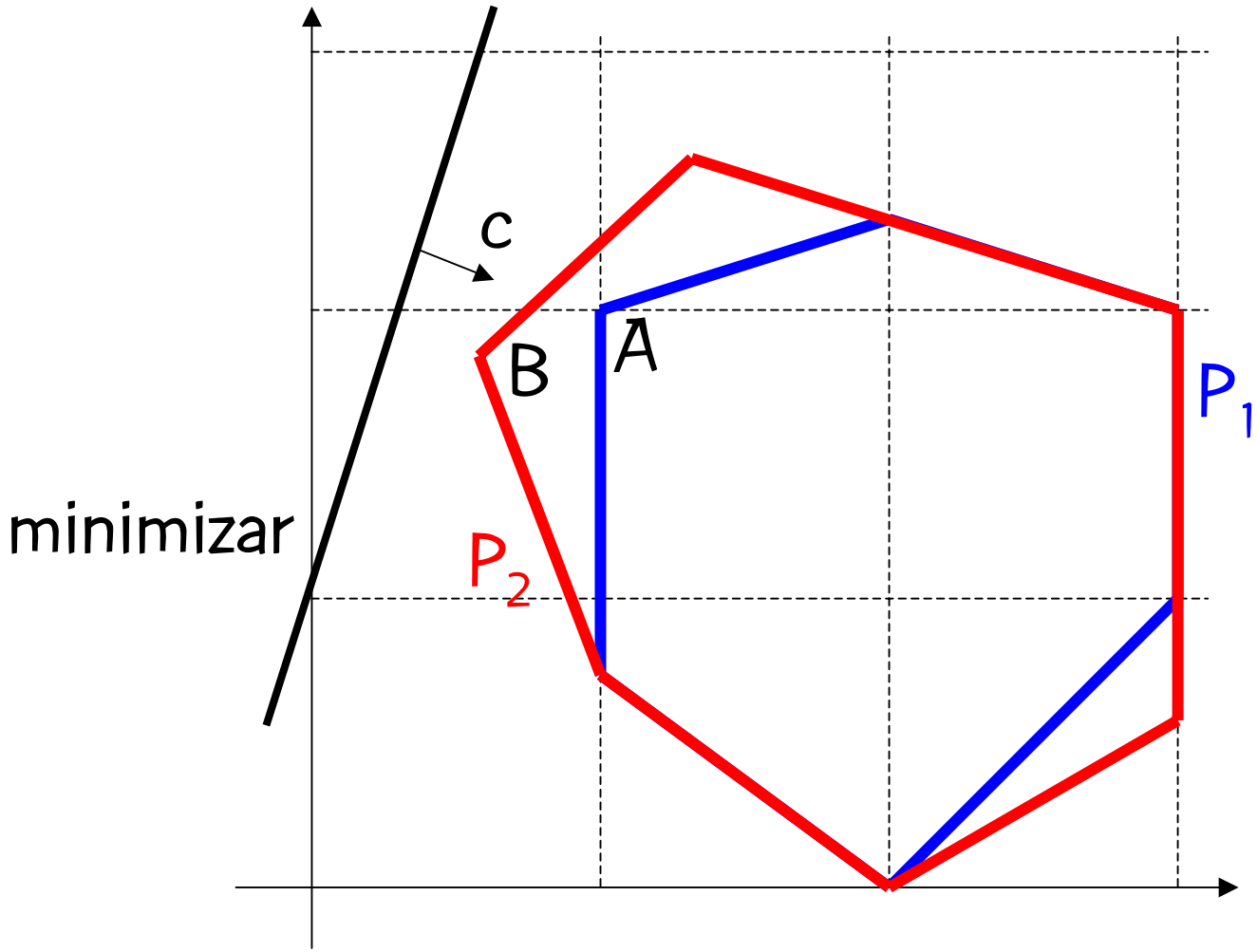
  - Com a restrição 2:

$$P_2 = \left\{(x,y) : \sum_{j=1}^{n} x_{ij} = 1, \forall i; \sum_{i=1}^{m} x_{ij} \le m.y_j, \forall j; 0 \le x_{ij} \le 1,\ 0 \le y_j \le 1, \forall i,j\right\}$$

Formulação de problemas

- Observa-se que $P_1 \subseteq P_2$: o conjunto de soluções da relaxação linear de P_1 é mais próximo do conjunto de soluções inteiras do que o conjunto de soluções da relaxação linear de P_2.
- Sejam z^* o valor da solução ótima do problema de programação inteira original e z_1 e z_2 os valores ótimos das duas relaxações: $z_2 \leq z_1 \leq z^*$

Formulação de problemas



Formulação de problemas

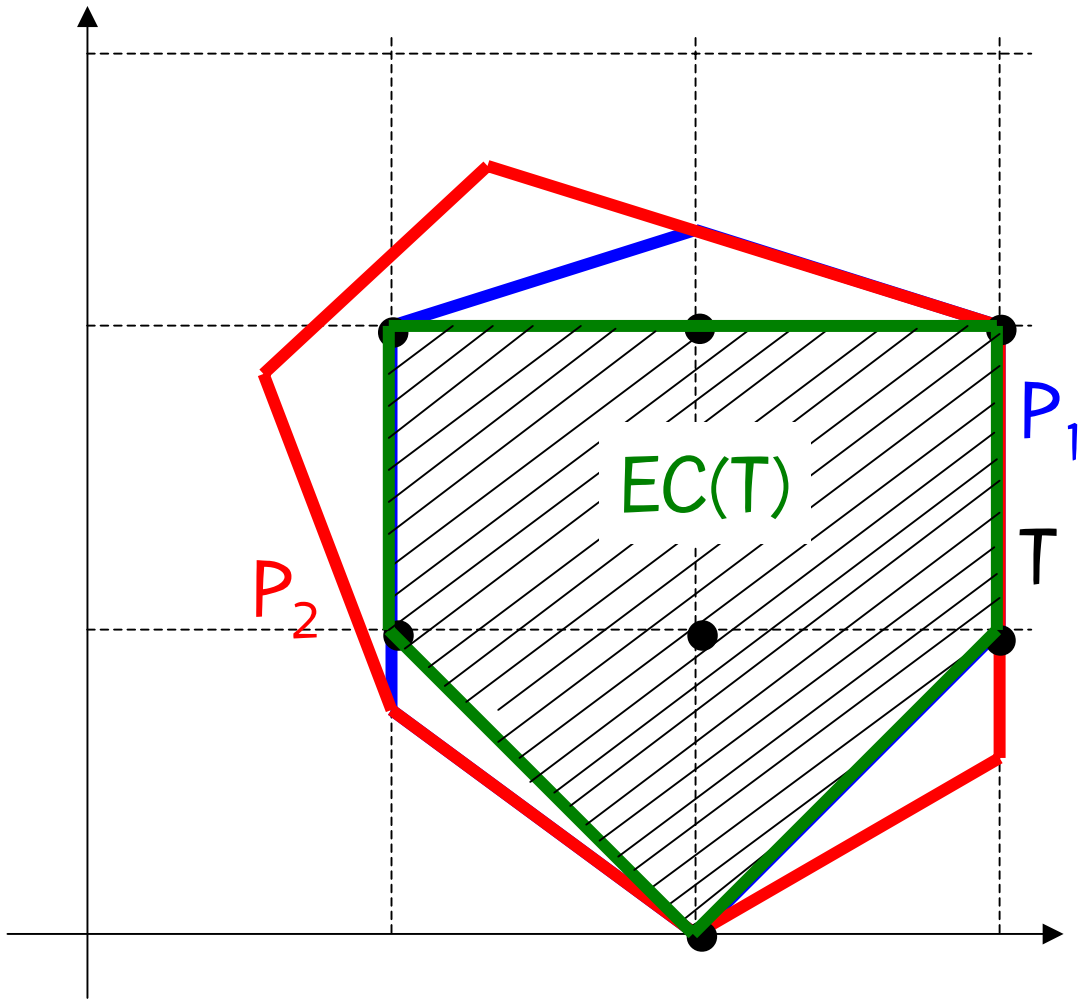
- Muitos métodos para a solução de problemas de programação inteira dependem de bons limites inferiores para z^*, tais como z_1 ou z_2.
- Quanto mais próximo este limite inferior estiver de z^*, melhor será a formulação.
- Neste caso, a primeira formulação é melhor mesmo tendo um maior número de restrições.
- Qual seria a melhor (ideal) formulação possível para um problema de programação inteira?

Formulação de problemas

- Seja $T = \{x^1, x^2, ..., x^k\}$ o conjunto de soluções inteiras de um problema de programação linear inteira.
- Supõe-se que este conjunto seja finito, já que por hipótese o poliedro P definido pelas restrições lineares é limitado.
- Seja ainda EC(T) a envoltória convexa de T, isto é, o "menor" conjunto convexo tal que $T \subseteq EC(T)$.
- Todos os pontos extremos de EC(T) são inteiros.
- Ademais, $EC(T) \subseteq P$.
- Suponha que EC(T) possa ser conhecido explicitamente:

$$EC(T) = \{x : D.x \leq d\}$$

Formulação de problemas



Formulação de problemas

- Então, o problema de programação inteira

$$\text{minimizar } c.x$$

$$\text{sujeito a : } x \in P, x \text{ inteiro}$$

é equivalente a

$$\text{minimizar } c.x$$

$$\text{sujeito a : } x \in T,$$

que também é equivalente ao problema de programação linear

$$\text{minimizar } c.x$$

$$\text{sujeito a : } x \in EC(T).$$

Formulação de problemas

- Considerando-se nossa habilidade para resolver problemas de programação linear, seria interessante conseguir sempre uma formulação cujo conjunto de soluções viáveis de sua relaxação linear fosse igual à envoltória convexa das soluções inteiras.
- Difícil, para poucos problemas isto é possível!
- Procurar obter uma formulação cujo conjunto de soluções viáveis de sua relaxação linear seja o mais próximo possível da envoltória convexa das soluções inteiras.

Formulação de problemas

- Conclusão:
  - A qualidade de uma formulação de um problema de programação inteira cujo conjunto de soluções viáveis é T pode ser avaliada pela proximidade do conjunto de soluções viáveis de sua relaxação linear com a envoltória convexa de T.
  - Dadas duas formulações A e B para um problema de programação inteira, cujas relaxações lineares possuem P_A e P_B, respectivamente, como seus conjuntos de soluções viáveis, diz-se que a formulação A é pelo menos tão forte quanto a formulação B se $P_A \subseteq P_B$.
  - Uma formulação mais forte freqüentemente tem um número muito maior de variáveis e de restrições.

Métodos de solução

- Contrariamente à programação linear, problemas de programação inteira são difíceis de serem resolvidos.
- Não existe um algoritmo geral eficiente para a solução de um problema de programação inteira qualquer.
- De forma simplificada, define-se como eficiente um algoritmo cujo número de iterações é polinomial em alguma medida que descreva o tamanho do problema a ser resolvido: $\sim n^2$ (soma de matrizes), $\sim n^3$ (produto de matrizes), $\sim \log n$, $\sim (n \log n)$ (ordenação), $\sim (m+n)$, $\sim (m.n)$, etc.

Métodos de solução

- Métodos exatos
  - Métodos de planos de cortes
  - Métodos poliedrais
  - Branch-and-bound
  - Programação dinâmica (parte IV)
- Métodos aproximados
  - Algoritmos aproximativos
  - Heurísticas construtivas (parte III)
  - Busca local (parte III)
  - Metaheurísticas (parte III)

Métodos de solução

Métodos de planos de corte

- Problema de programação inteira (PI): minimizar $c.x$

  sujeito a :

  $A.x = b$

  $x \geq 0$ e inteiro

- Relaxação linear (RL): minimizar $c.x$

  sujeito a :

  $A.x = b$

  $x \geq 0$

Métodos de solução

- Idéia básica: resolver o problema de programação inteira através da solução de uma seqüência de problemas de programação linear:
  1. Resolva a relaxação linear de PI e seja x* sua solução ótima.
  2. Se x* é inteiro, então x* é a solução ótima do problema PI.
  3. Caso contrário, identificar uma desigualdade satisfeita por todas as soluções inteiras do problema PI, mas não por x*.
  4. Adicionar esta desigualdade à relaxação linear de PI e retornar ao passo (1) acima.

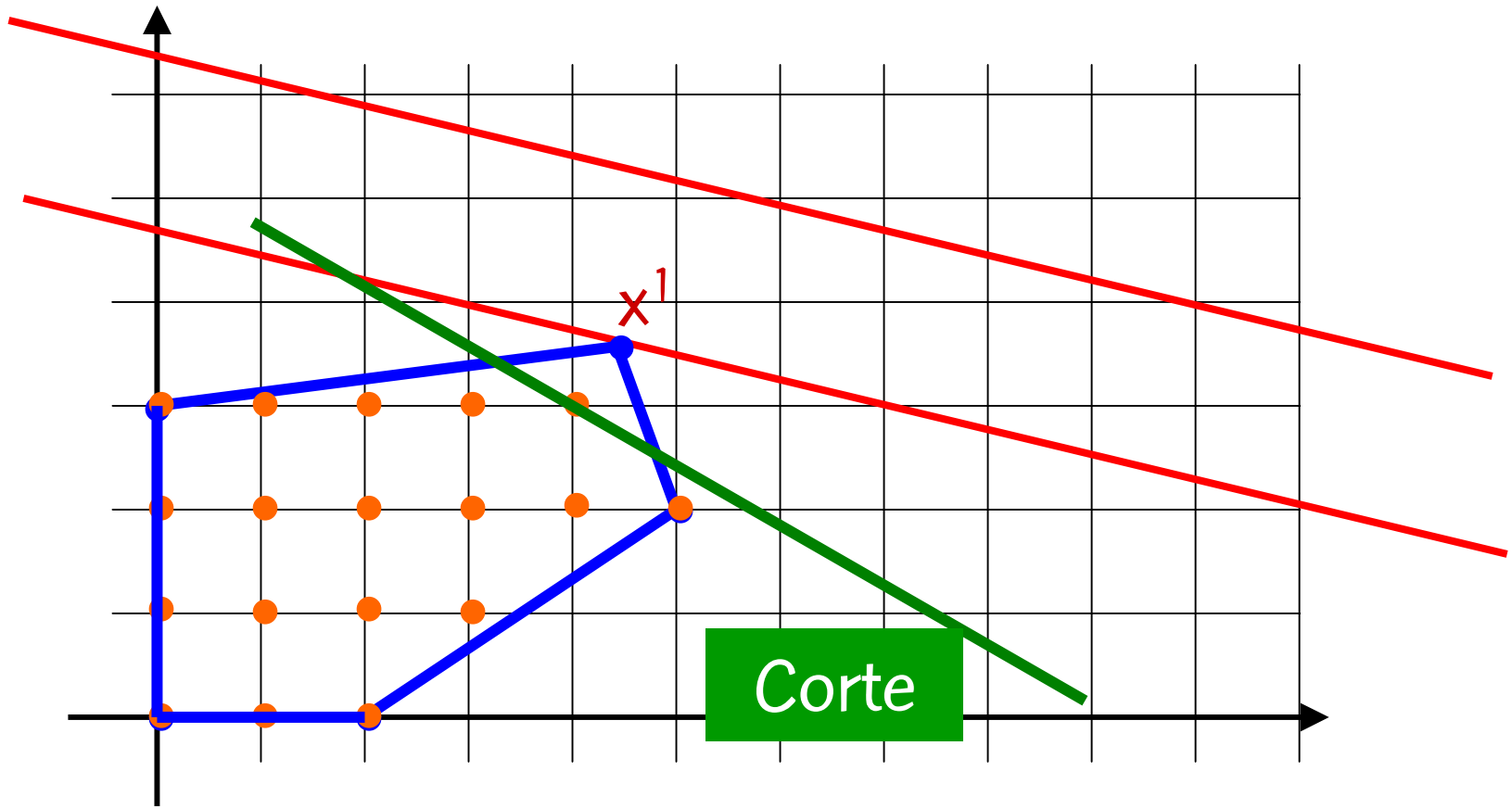
Métodos de solução

- Exemplo de um corte simples:
  - Seja x* a solução ótima da relaxação linear, na qual pelo menos uma variável é fracionária.
  - Seja N o conjunto dos índices das variáveis não-básicas.
  - Como esta solução é fracionária, ela não pode ser solução de PI.
  - Então, qualquer solução viável de PI deverá satisfazer à restrição adicional

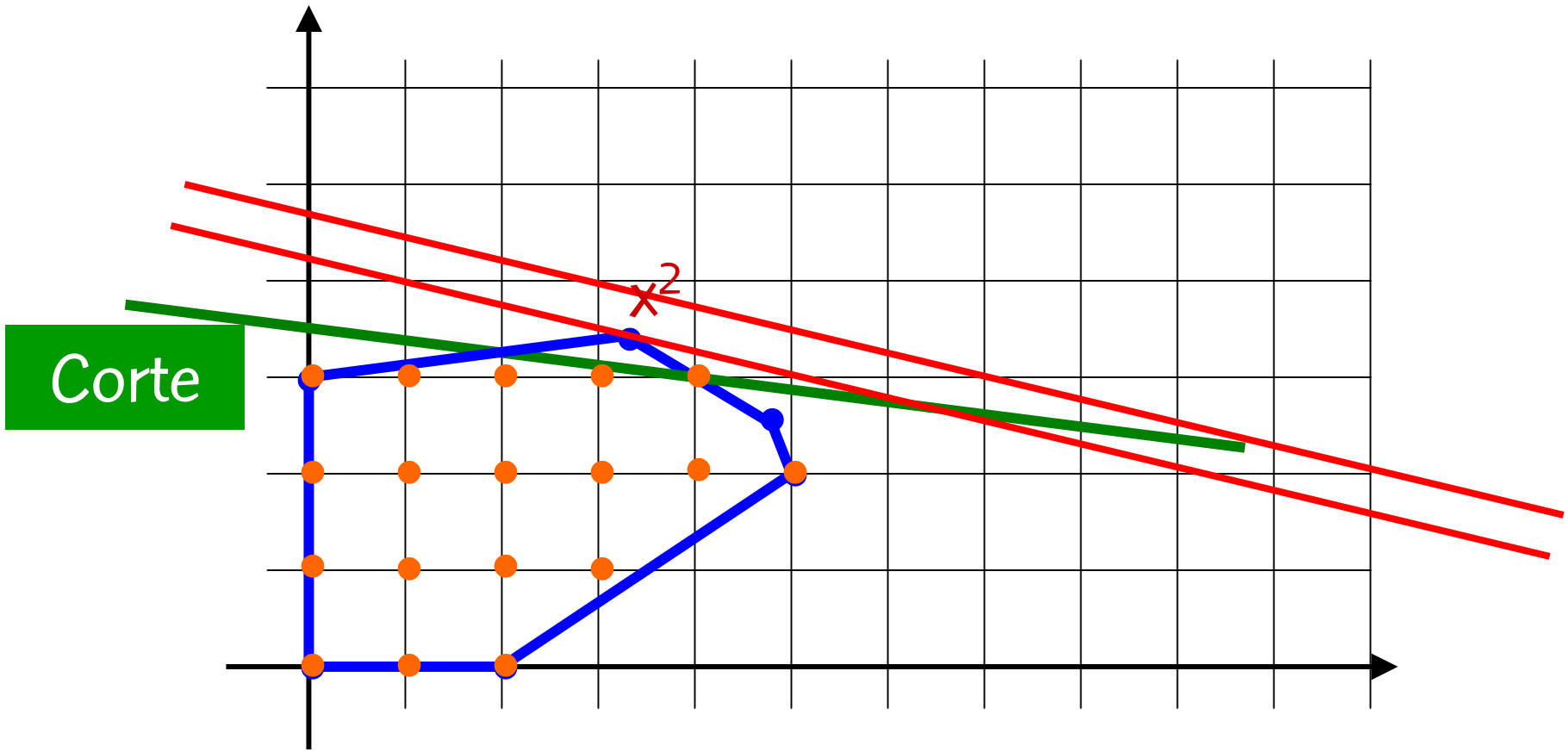
$$\sum_{j \in N} x_j \geq 1$$

(que é satisfeita por todas as soluções inteiras de PI, mas não por x*).

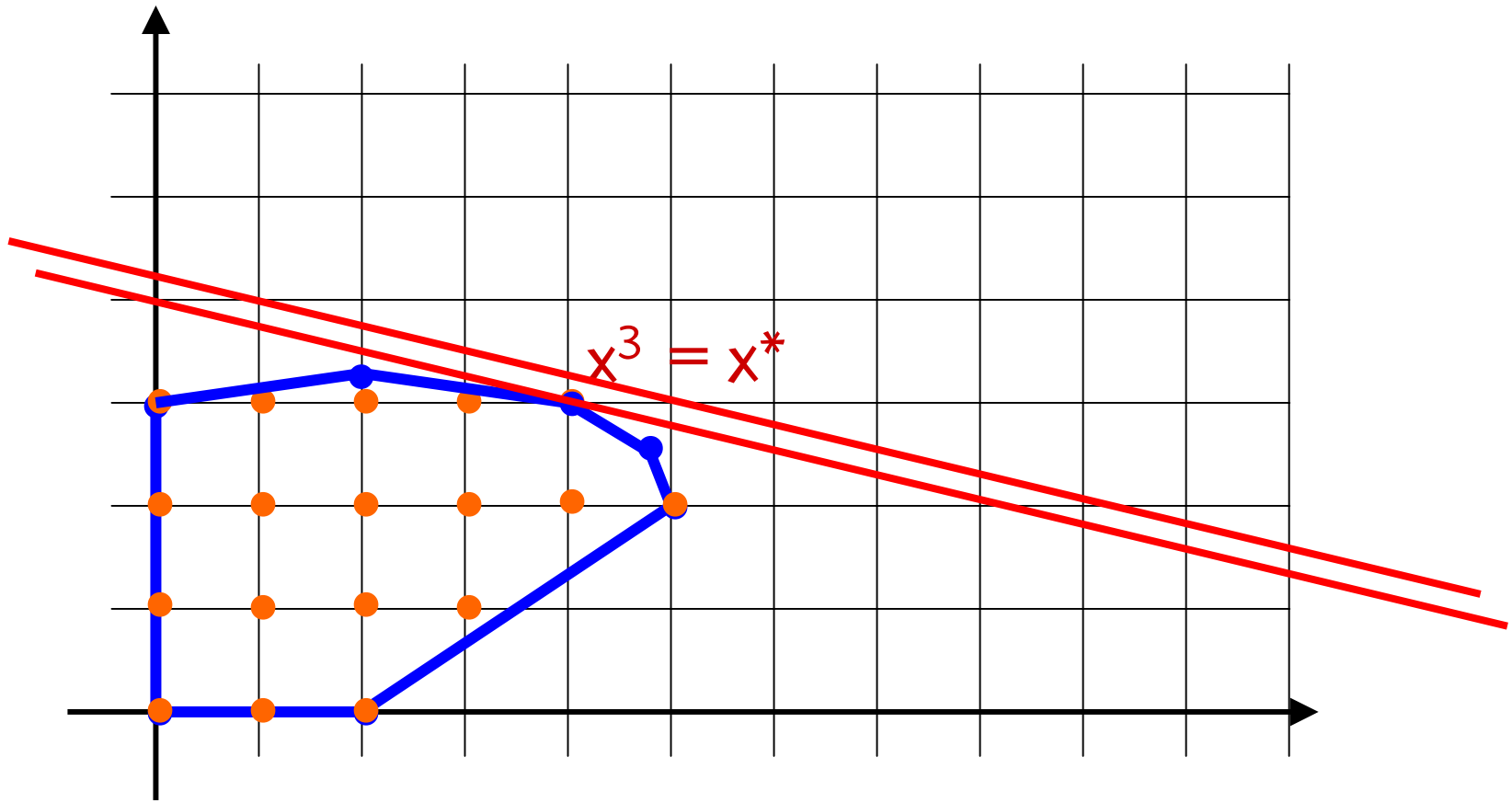
Métodos de solução



Métodos de solução

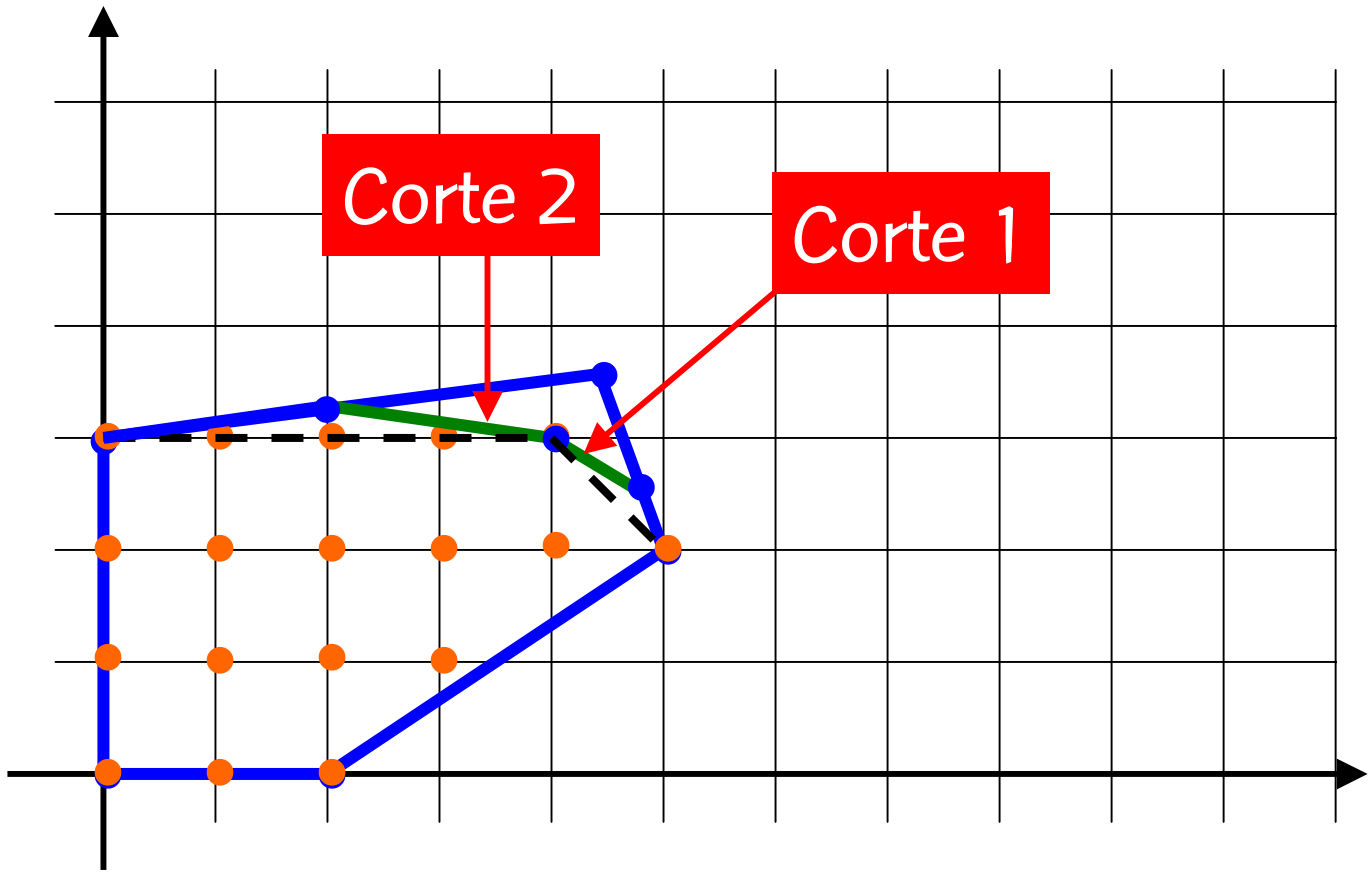


Métodos de solução



x* é a solução ótima do problema de programação inteira.

Métodos de solução



Aproximação da envoltória convexa na vizinhança do ótimo.

Métodos de solução

- Outros exemplos de cortes (derivação baseada nos resultados do algoritmo Simplex):
  - Cortes fracionários de Gomory
  - Cortes inteiros de Gomory
  - Cortes disjuntivos
  - Cortes de Benders
- Dificuldade: cortes genéricos eliminam uma fração muito pequena do poliedro que representa a região viável da relaxação linear e, por esta razão, seu desempenho é muito limitado (convergência muito lenta e problemas numéricos).

Métodos de solução

Métodos poliedrais

- Cortes baseados na estrutura algébrica de cada problema específico.
- Cortes mais profundos, que coincidem com as faces da envoltória convexa do conjunto de soluções viáveis inteiras.
- Dificuldade: a caracterização dos cortes depende da estrutura de cada problema a ser resolvido e estes nem sempre podem ser obtidos, envolvendo um longo e complexo estudo algébrico.

Métodos de solução

Branch-and-bound

- Estratégia do tipo "divisão-e-conquista" para explorar o conjunto de soluções viáveis inteiras, empregando limites inferiores (caso de minimização) para evitar a exploração de partes do conjunto de soluções viáveis inteiras.
- Problema de minimização:

minimizar $c.x$

sujeito a : $x \in P, x$ inteiro

- $P = P_1 \cup P_2 \cup \ldots \cup P_p$

Métodos de solução

Branch-and-bound

- Transformar o problema original em p subproblemas:

  minimizar $c.x$

  sujeito a : $x \in P_i, x$ inteiro, $\forall i = 1,..., p$

- Resolvê-los e salvar a melhor solução encontrada.
- Cada subproblema pode ser tão difícil quanto o problema inicial.
- Proceder recursivamente por decomposição, gerando subproblemas progressivamente menores até que possam ser resolvidos de forma exata.

Métodos de solução

- Assume-se a existência de uma forma simples de calcular um limite inferior para o valor ótimo de cada subproblema:
  - O limite inferior é usado para guiar a exploração dos subproblemas ainda em aberto.
  - Estratégia simples para calcular um limite inferior?
  - Resolver a relaxação linear!
- Heurísticas ou resolução exata de algum subproblema:
  - Limite superior UB para o custo da solução ótima = = custo da melhor solução viável inteira já encontrada

Métodos de solução

- Essência do método: se o limite inferior correspondente a um subproblema S_i é maior ou igual a UB, então este subproblema pode ser descartado.
  - Sua solução ótima inteira nunca poderá ser melhor do que a melhor solução inteira já conhecida, pois seu custo será maior ou igual a UB.
- Manter em memória:
  - Lista de subproblemas ativos ou abertos, ainda não resolvidos.
  - Custo UB da melhor solução viável inteira conhecida.

Métodos de solução

1. Inicializar a lista de subproblemas ativos com o original.
2. Selecionar um subproblema ativo S_i e retirá-lo da lista.
3. Se S_i é inviável, descartá-lo. Senão, resolver sua relaxação e obter um limite inferior LB_i para seu valor.
4. Se $LB_i \geq UB$, descartar S_i.
5. Caso contrário (isto é, se $LB_i < UB$):
   - ou resolver S_i de forma exata,
   - ou decompor S_i em subproblemas que são inseridos na lista.
6. Se a lista não estiver vazia, retornar ao passo (2) acima.

Métodos de solução

- Estratégias em aberto no esquema anterior:
  - Como escolher um dos problemas em aberto no passo 2?
    - Busca em profundidade
    - Busca em amplitude
    - Busca pelo melhor limite inferior
  - Como obter um limite inferior LB_i no passo 3?
    - Estratégia mais freqüente: resolver a relaxação linear
    - Quanto melhor o limite inferior (isto é, quanto mais próximo do valor exato), mais rápido o algoritmo esgota a lista de subproblemas.
  - Como decompor um problema em subproblemas no passo 5?
    - Estratégia mais freqüente: adicionar restrições $x_i \leq \lfloor x_i^* \rfloor$ e $x_i \geq \lceil x_i^* \rceil$

Métodos de solução

- Exemplo 1:

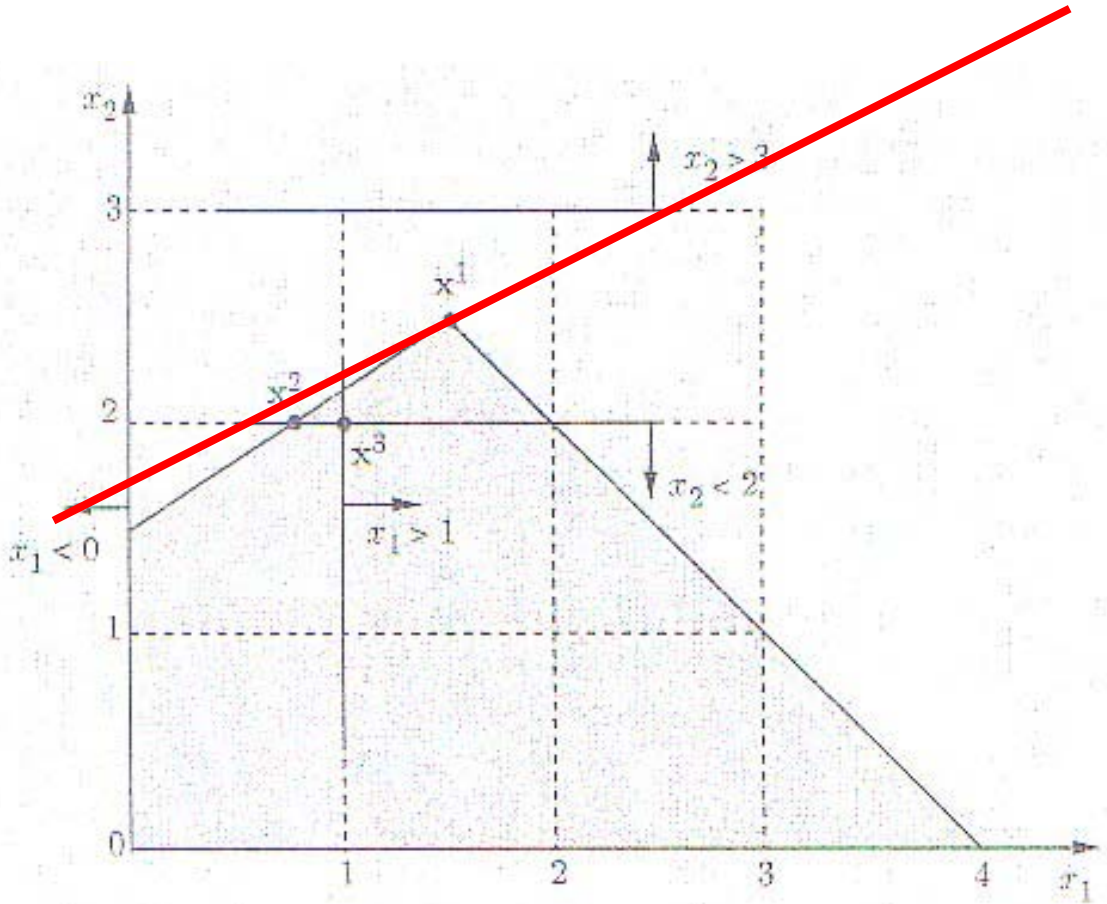
minimizar $x_1 - 2x_2$

sujeito a : $-4x_1 + 6x_2 \leq 9$

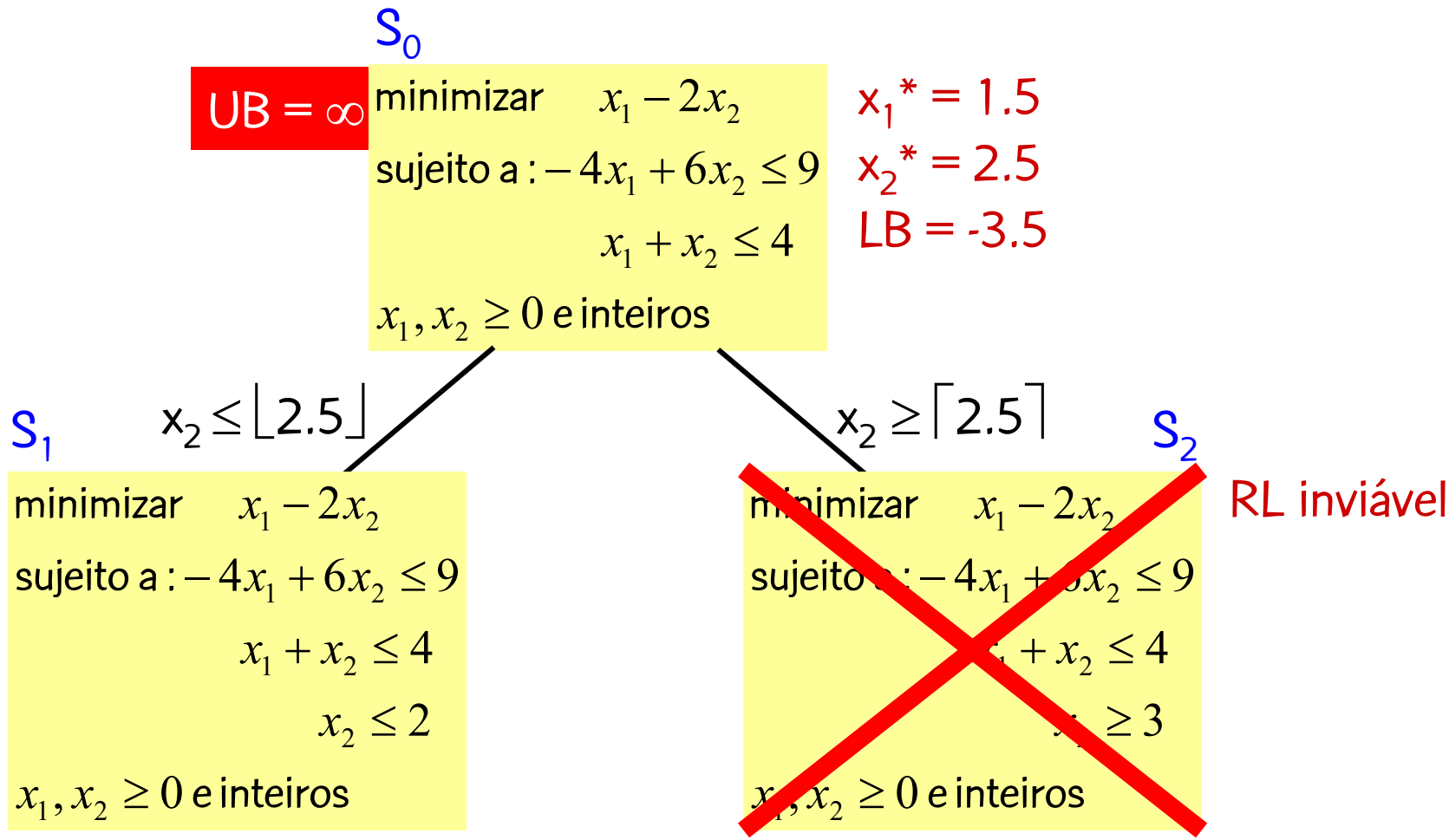
$x_1 + x_2 \leq 4$

$x_1, x_2 \geq 0$

x_1, x_2 inteiros



Métodos de solução



Métodos de solução

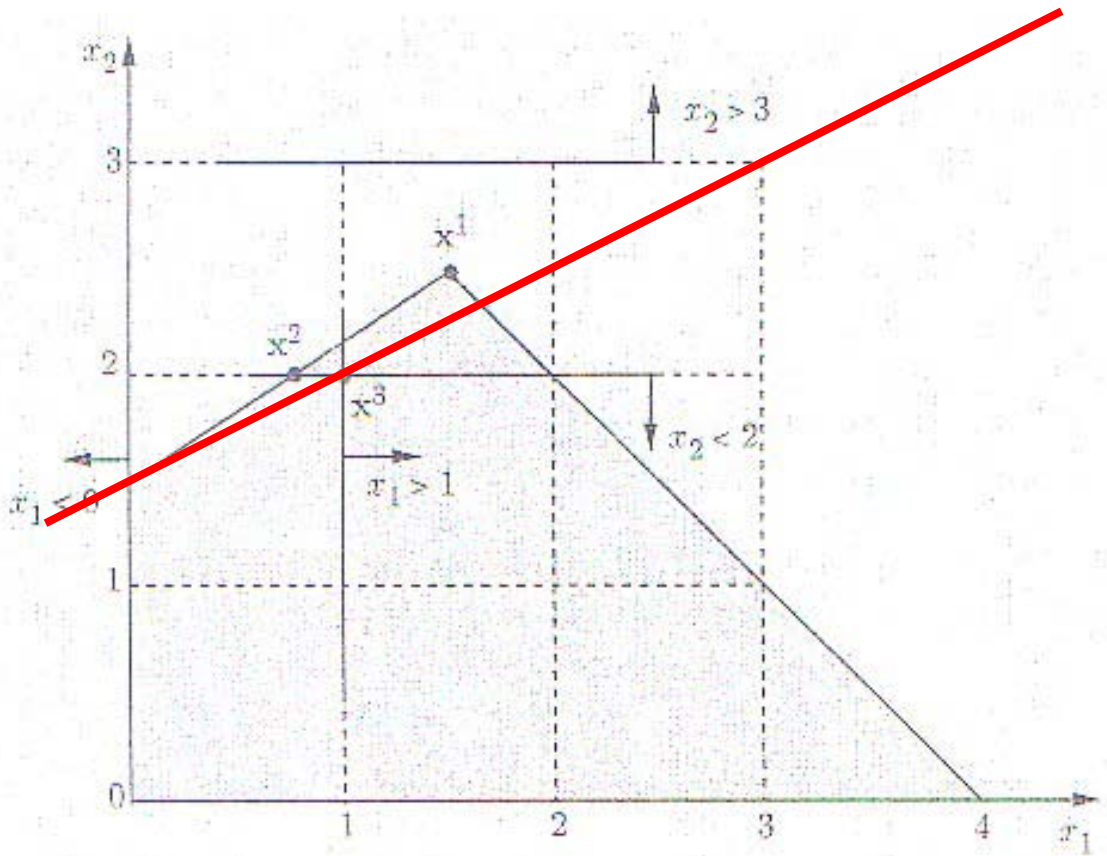
minimizar $x_1 - 2x_2$

sujeito a : $-4x_1 + 6x_2 \leq 9$

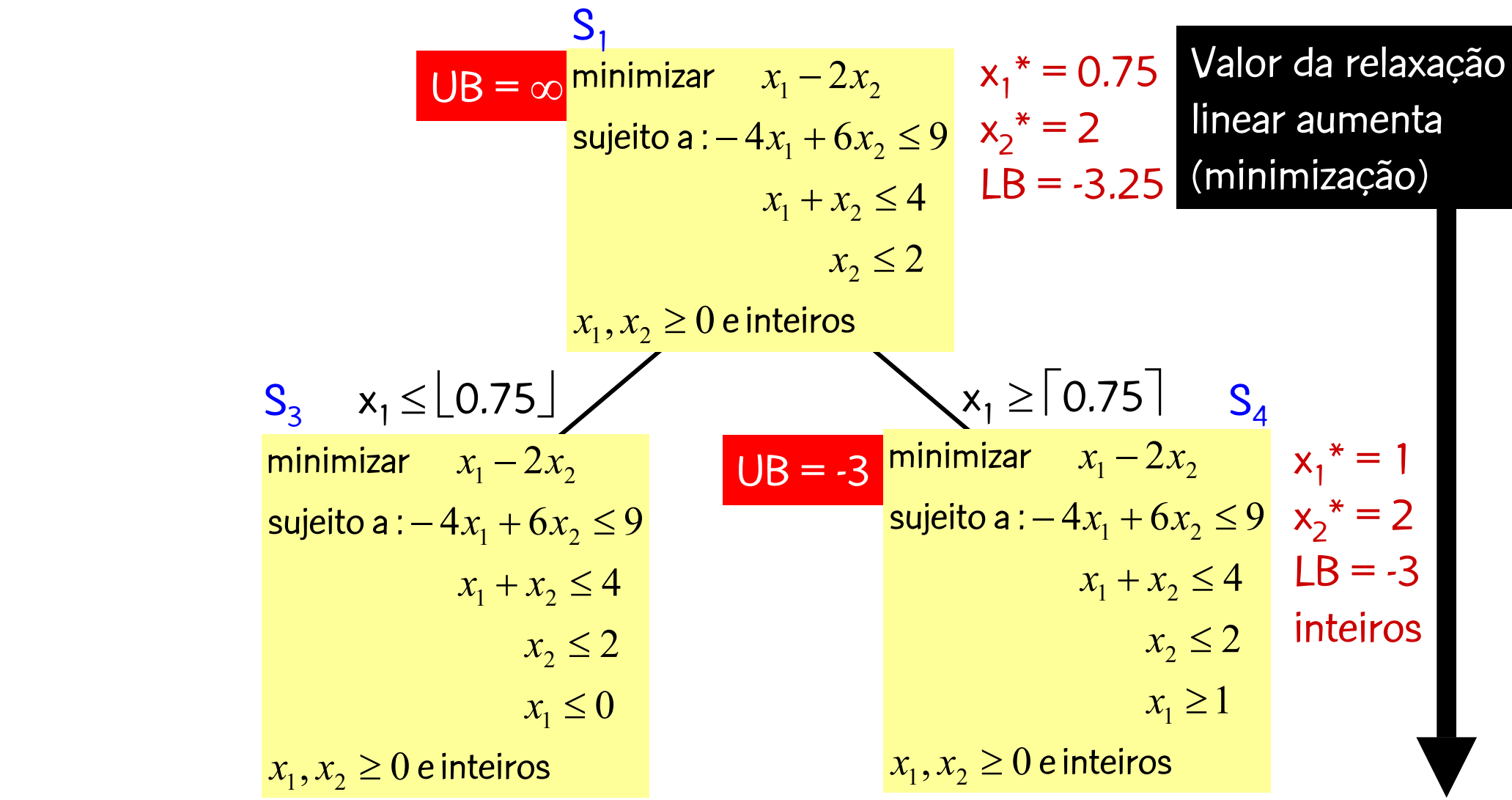
$x_1 + x_2 \leq 4$

$x_1, x_2 \geq 0$

x_1, x_2 inteiros



Métodos de solução



Métodos de solução

Lista de subproblemas ativos vazia!

S_1

minimizar $x_1 - 2x_2$

sujeito a : $-4x_1 + 6x_2 \le 9$

$x_1 + x_2 \le 4$

$x_2 \le 2$

$x_1, x_2 \ge 0$ e inteiros

$x_1^* = 0.75$

$x_2^* = 2$

LB = -3.25

$x_1 \le \lfloor 0.75 \rfloor$

S_3

minimizar $x_1 - 2x_2$

sujeito a : $-4x_1 + 6x_2 \le 9$

$x_1 + x_2 \le 4$

$x_2 \le 2$

$x_1 \le 0$

$x_1, x_2 \ge 0$ e inteiros

$x_1^* = 0$

$x_2^* = 3/2$

LB = -3

LB ≥ UB

$x_1 \ge \lceil 0.75 \rceil$

S_4

minimizar $x_1 - 2x_2$

sujeito a : $-4x_1 + 6x_2 \le 9$

$x_1 + x_2 \le 4$

$x_2 \ge 3$

$x_1 \ge 1$

$x_1, x_2 \ge 0$ e inteiros

UB = -3

$x_1^* = 1$

$x_2^* = 2$

LB = -3

inteiros

Métodos de solução

Lista de subproblemas ativos vazia!

S_1

minimizar $x_1 - 2x_2$

sujeito a : $-4x_1 + 6x_2 \le 9$

$x_1 + x_2 \le 4$

$x_2 \le 2$

$x_1, x_2 \ge 0$ e inteiros

$x_1^* = 0.75$

$x_2^* = 2$

LB = -3.25

Solução ótima inteira

$x_1^* = 1$

$x_2^* = 2$

LB = -3

inteiros

$x_1 \le \lfloor 0.75 \rfloor$

S_3

minimizar $x_1 - 2x_2$

sujeito a : $-4x_1 + 6x_2 \le 9$

$x_1 + x_2 \le 4$

$x_2 \le 2$

$x_1 \le 0$

$x_1, x_2 \ge 0$ e inteiros

$x_1^* = 0$

$x_2^* = 3/2$

LB = -3

LB ≥ UB

$x_1 \ge \lceil 0.75 \rceil$

S_4

minimizar $x_1 - 2x_2$

sujeito a : $-4x_1 + 6x_2 \le 9$

$x_1 + x_2 \le 4$

$x_2 \ge 3$

$x_1 \ge 1$

$x_1, x_2 \ge 0$ e inteiros

UB = -3

Métodos de solução

- A escolha do subproblema ativo com o menor limite superior no passo 2 é justificada pela possibilidade de encontrar melhores soluções inteiras.
- Escolha da variável sobre a qual será feita a ramificação no passo 5:
  - Variável mais próxima de um valor inteiro
  - Variável mais afastada de um valor inteiro
- Não existem critérios sistematicamente melhores:
  - Ajustar os parâmetros de uma implementação de um algoritmo de branch-and-bound é uma tarefa difícil!

Métodos de solução

- Formulações mais justas dão melhores limites inferiores:
  - A resolução das relaxações lineares é mais demorada, mas...
  - ... melhores limites inferiores permitem descartar mais subproblemas e reduzir o número de nós da árvore!
- Problemas de tempo e de memória (crescimento da árvore)
- Resolver problemas de programação inteira é uma tarefa difícil e ingrata.
- Até mesmo pequenas alterações nos dados de entrada podem inviabilizar computacionalmente a resolução de uma determinada instância.

Métodos de solução

- Exemplo 2:

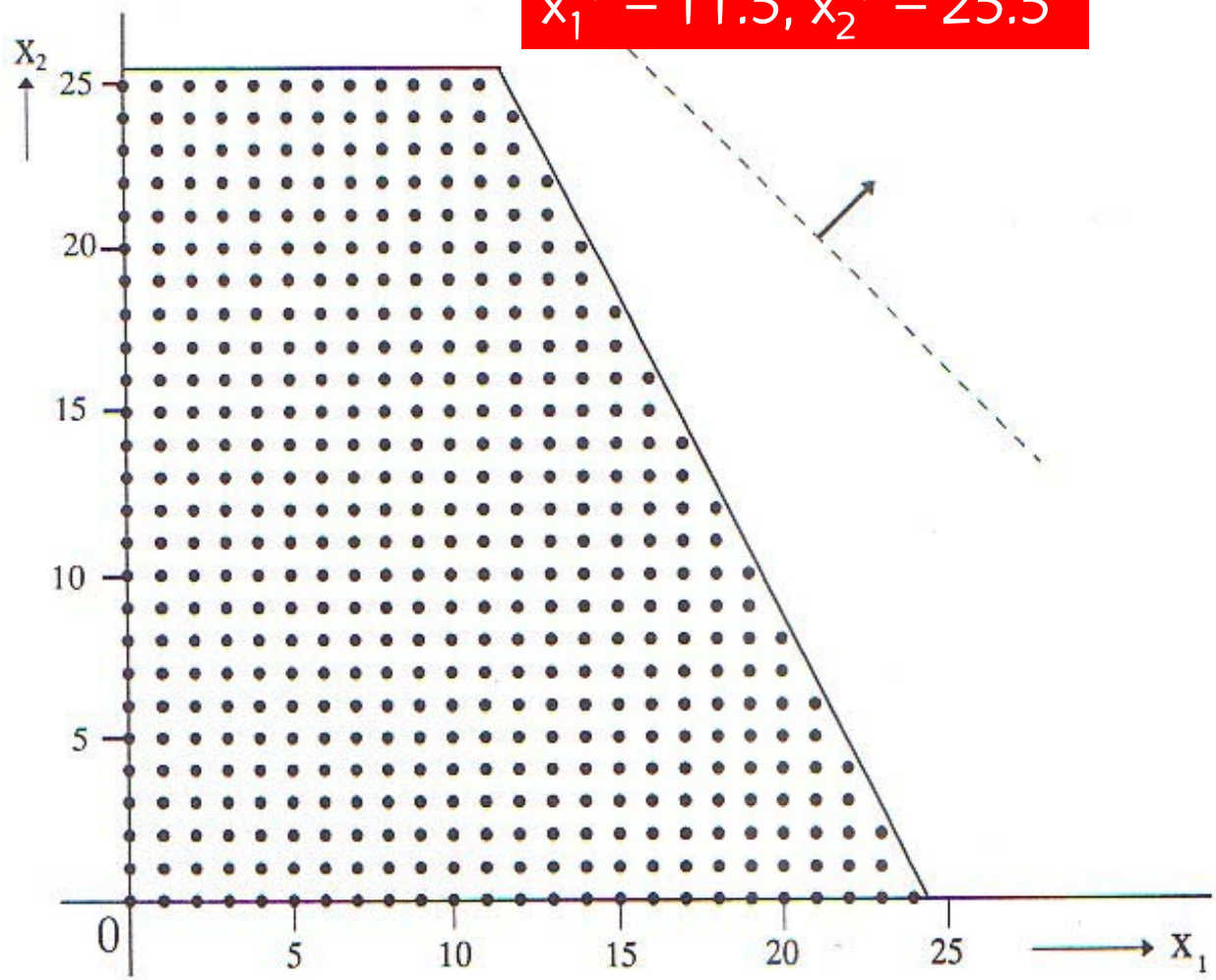
maximizar $1000x_1 + 700x_2$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros

$x_1^* = 11.5, x_2^* = 25.5$



Métodos de solução

maximizar $1000x_1 + 700x_2$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros

LB = -∞

z = 29350

$x_1 = 11\frac{1}{2}$, $x_2 = 25\frac{1}{2}$

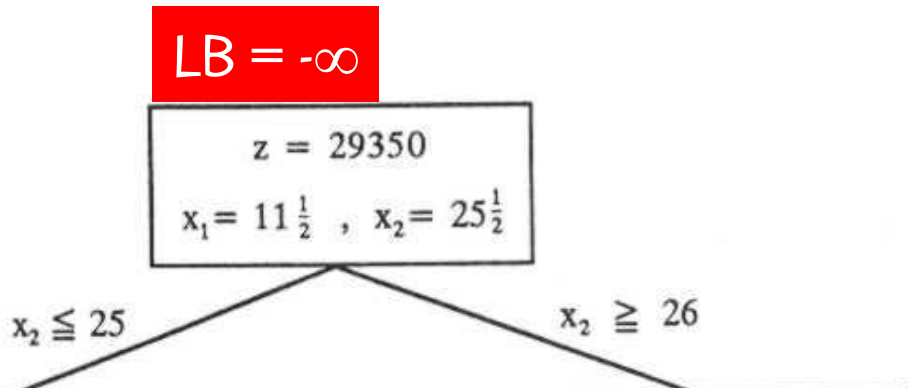
Métodos de solução

maximizar $1000x_1 + 700x_2$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros



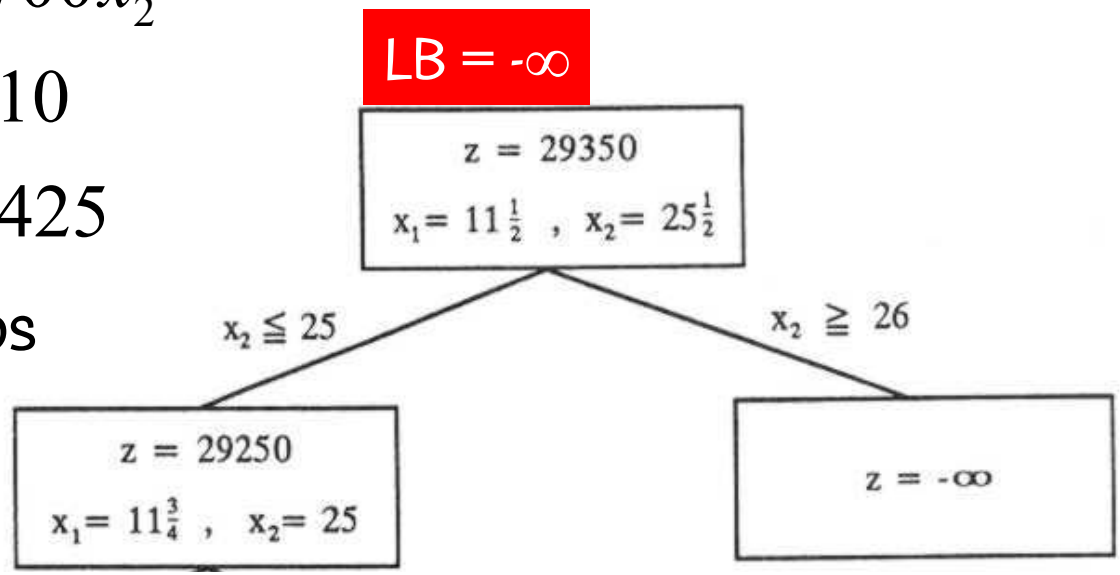
Métodos de solução

maximizar $1000x_1 + 700x_2$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros



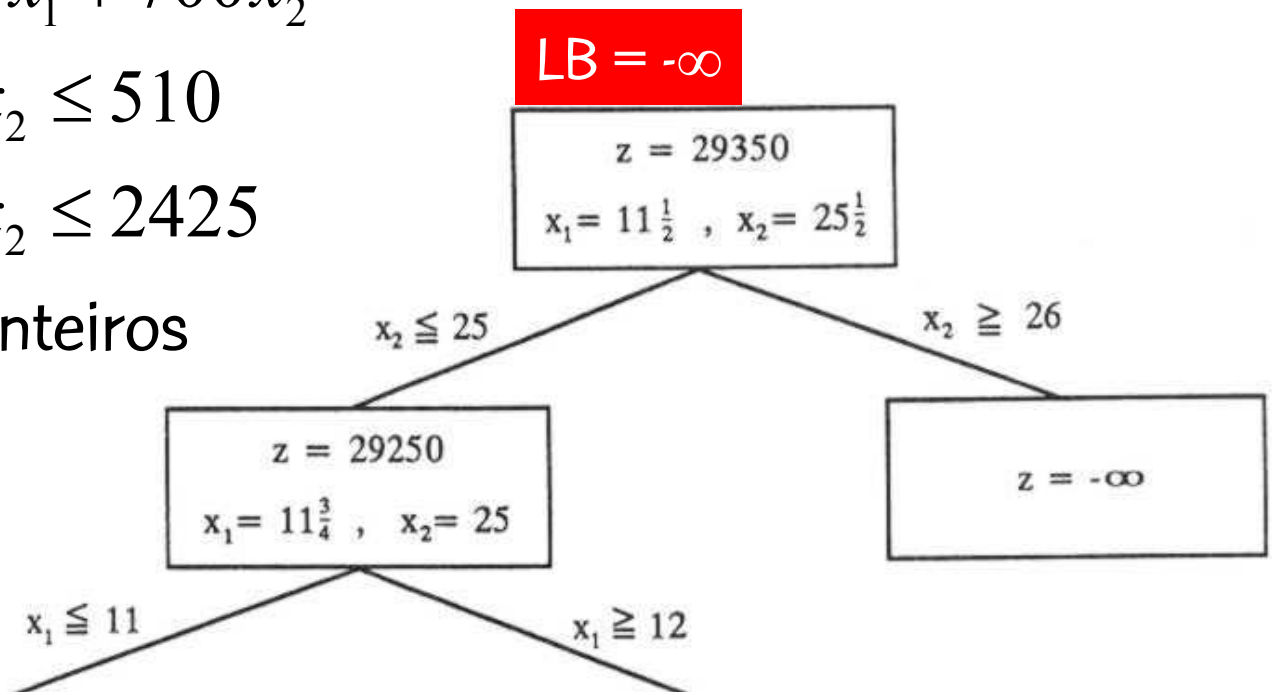
Métodos de solução

maximizar $1000x_1 + 700x_2$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros



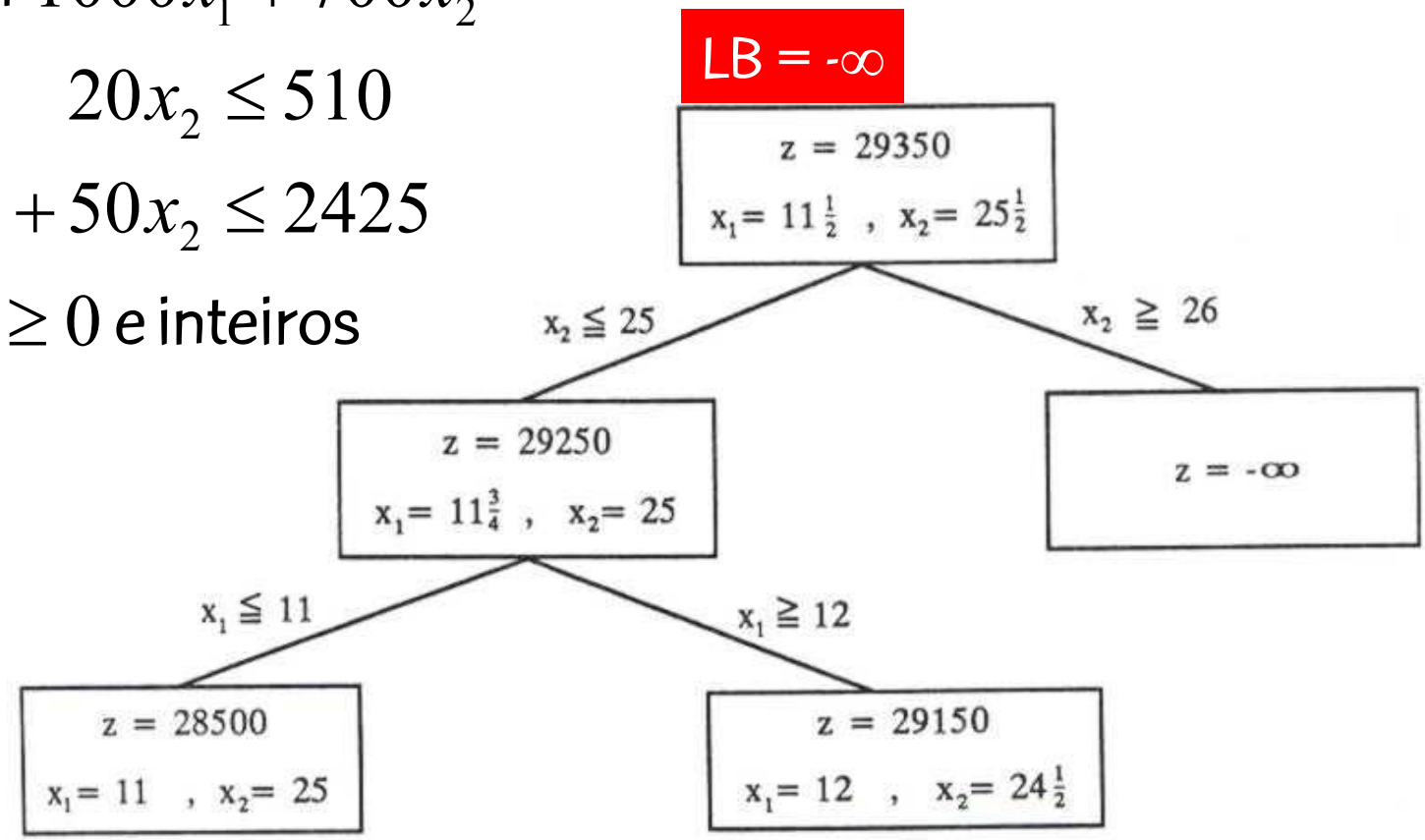
Métodos de solução

maximizar $1000x_1 + 700x_2$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros



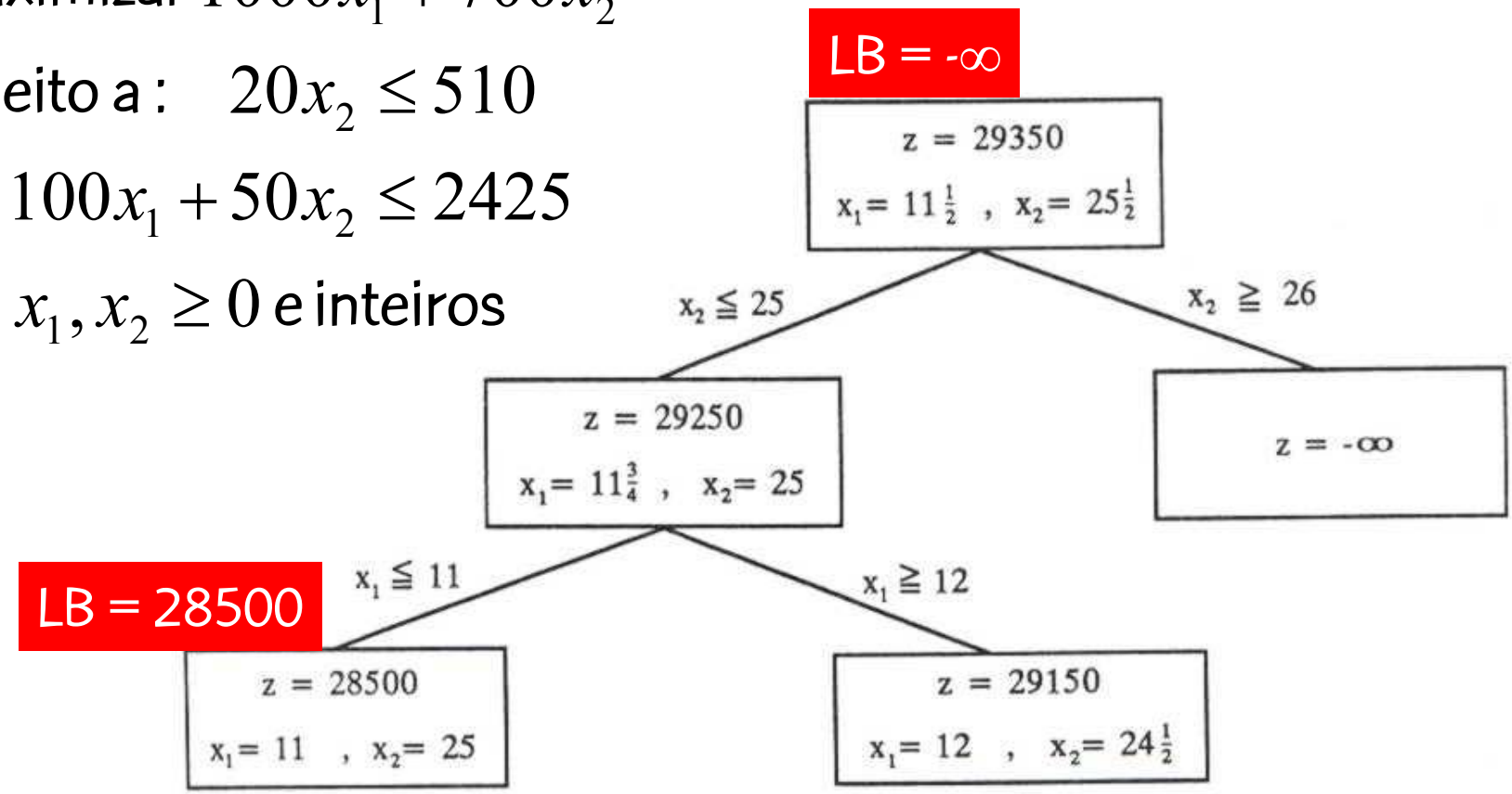
Métodos de solução

maximizar $1000x_1 + 700x_2$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros



Métodos de solução

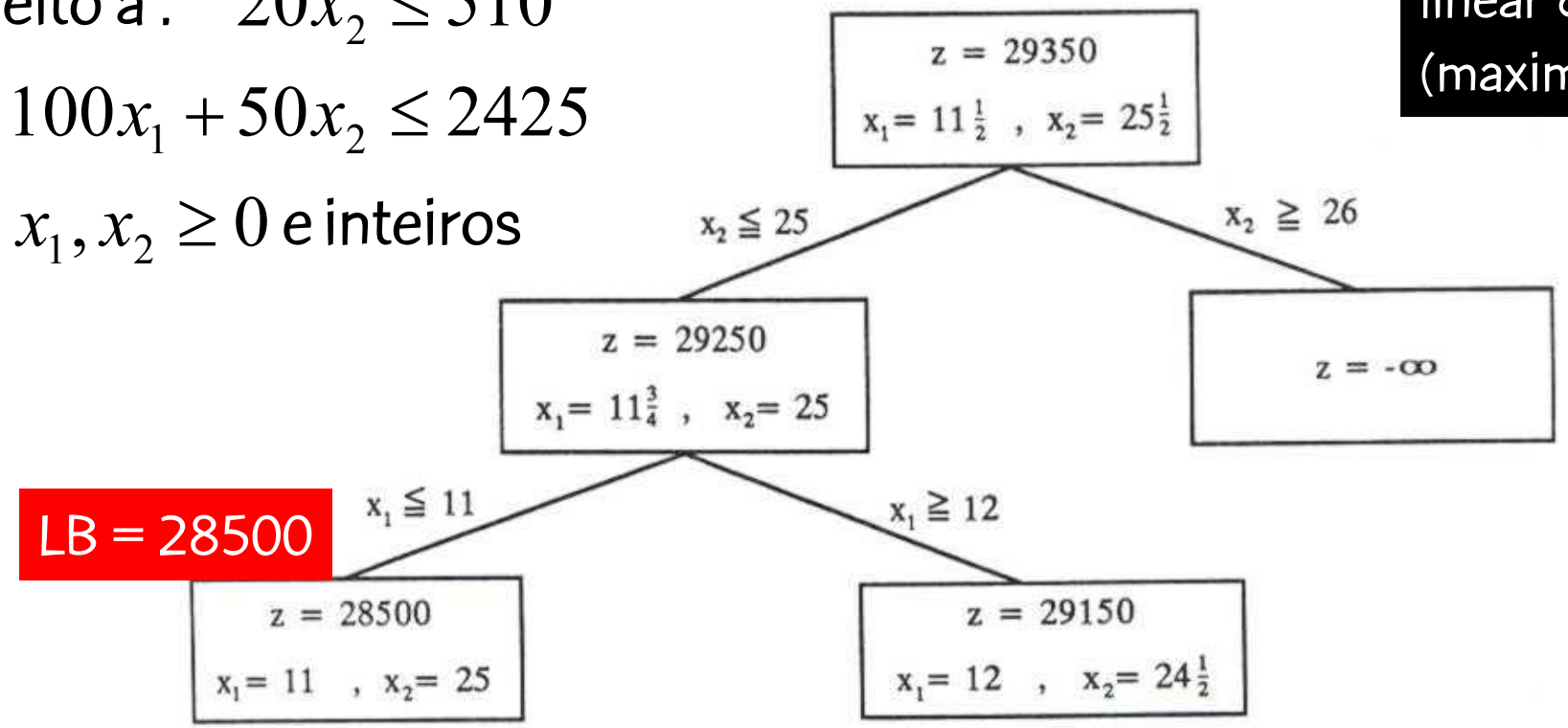
maximizar $1000x_1 + 700x_2$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros

Valor da relaxação linear diminui (maximização)



Métodos de solução

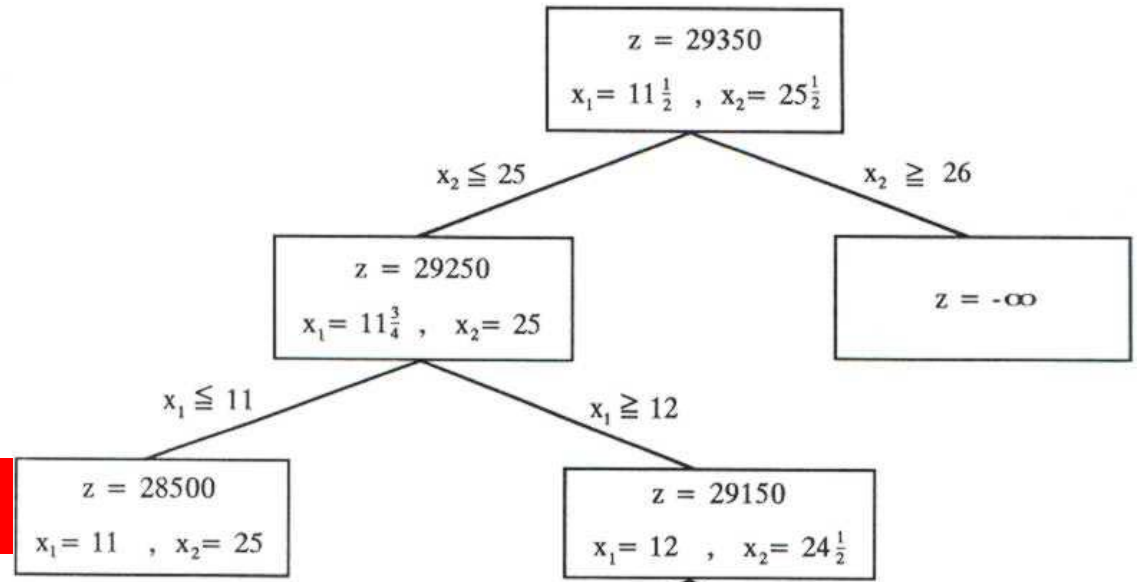
maximizar $1000x_1 + 700x$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros

LB = 28500



Métodos de solução

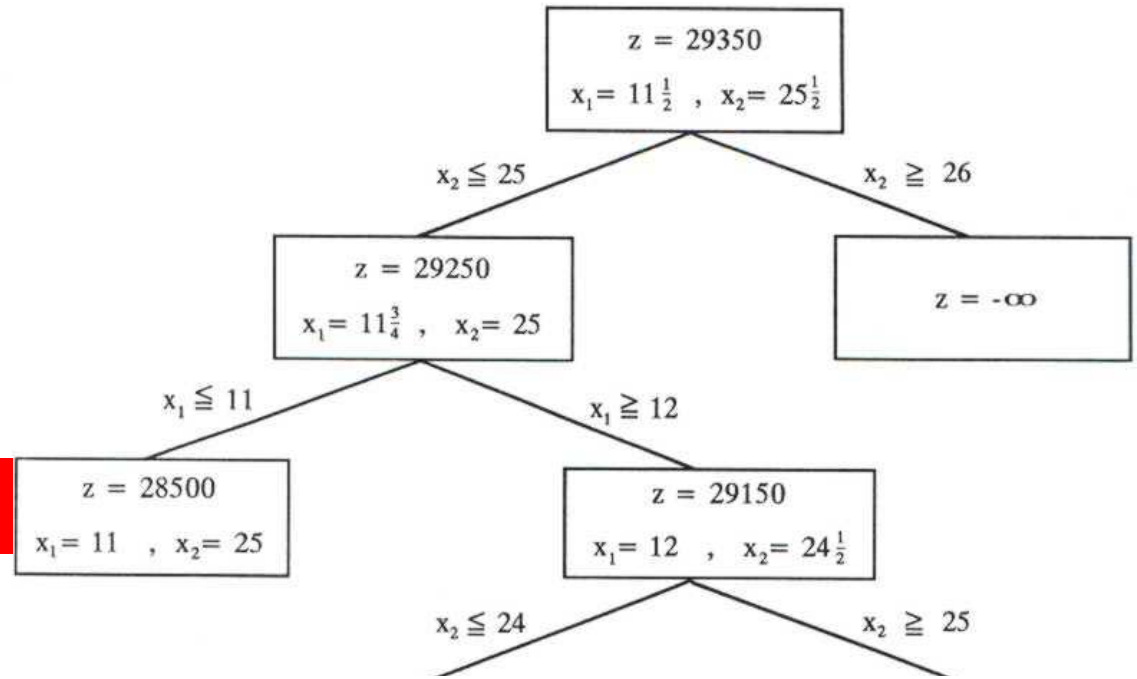
maximizar $1000x_1 + 700x$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros

LB = 28500



Métodos de solução

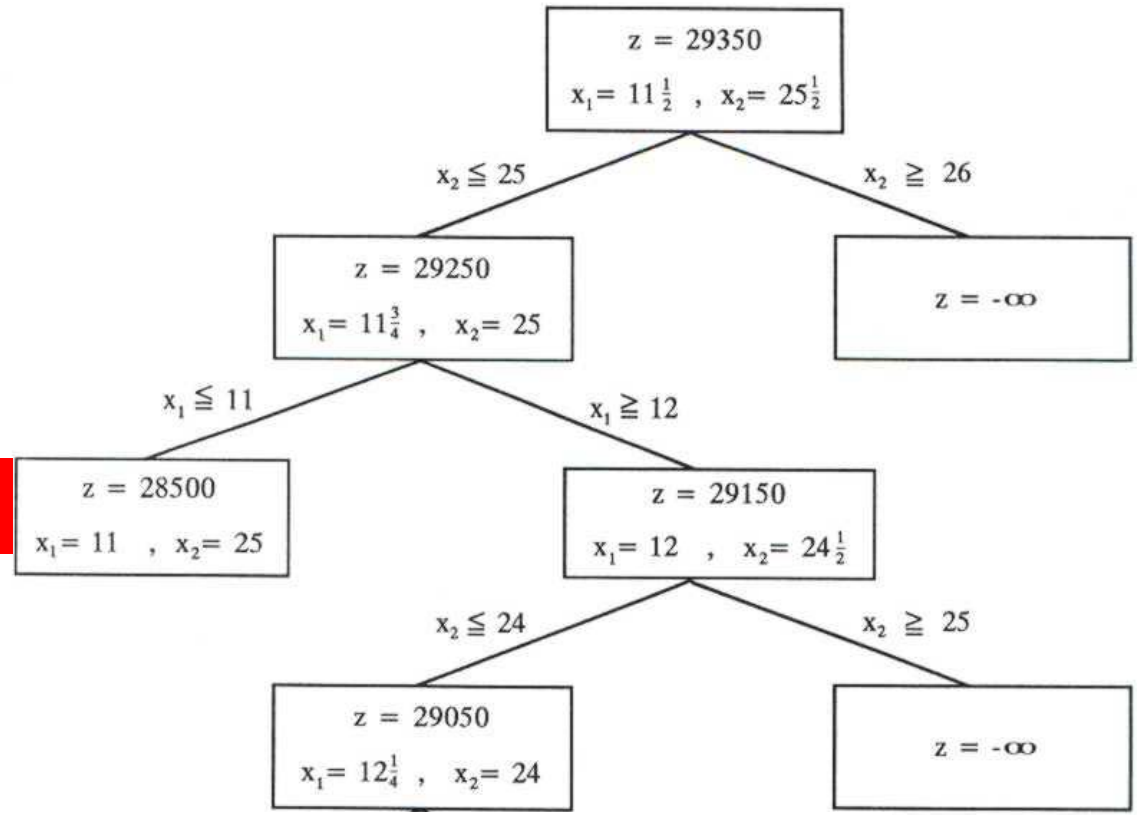
maximizar $1000x_1 + 700x$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros

LB = 28500



Métodos de solução

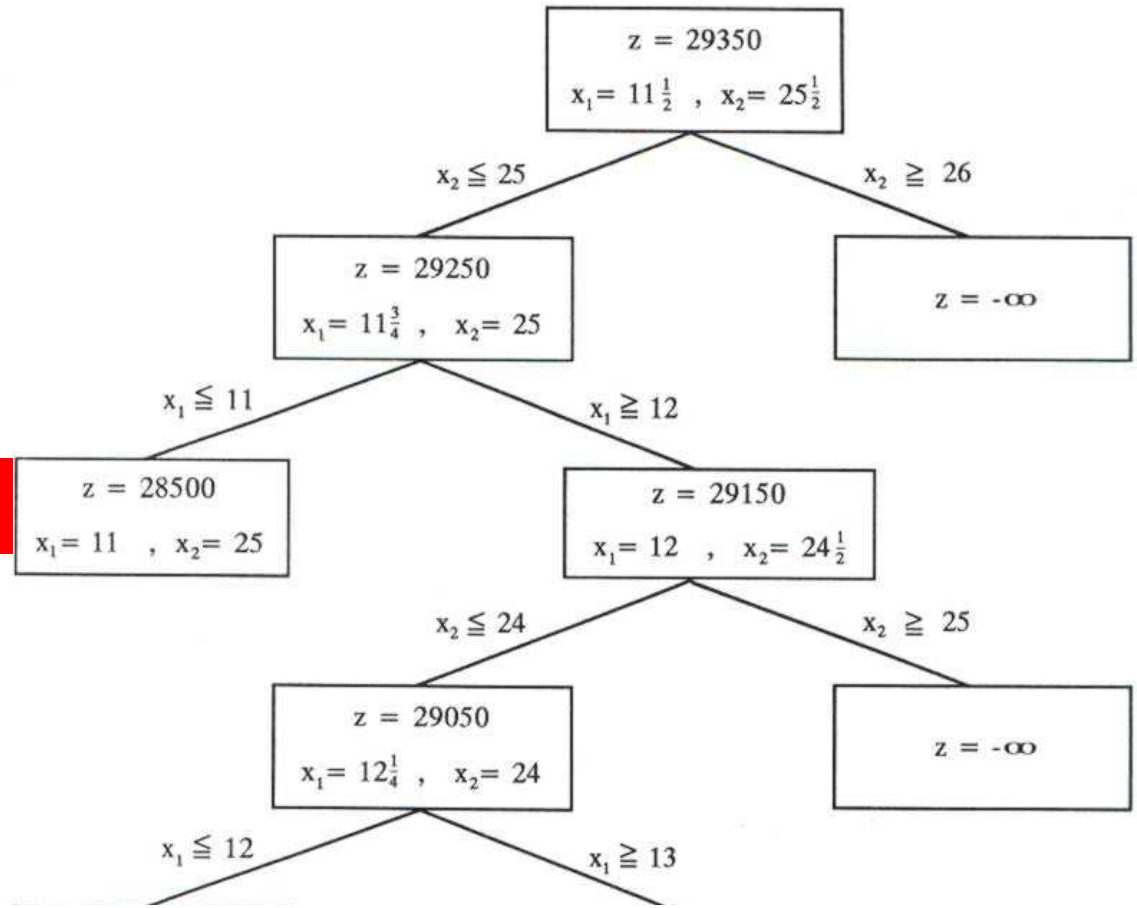
maximizar $1000x_1 + 700x$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros

LB = 28500



Métodos de solução

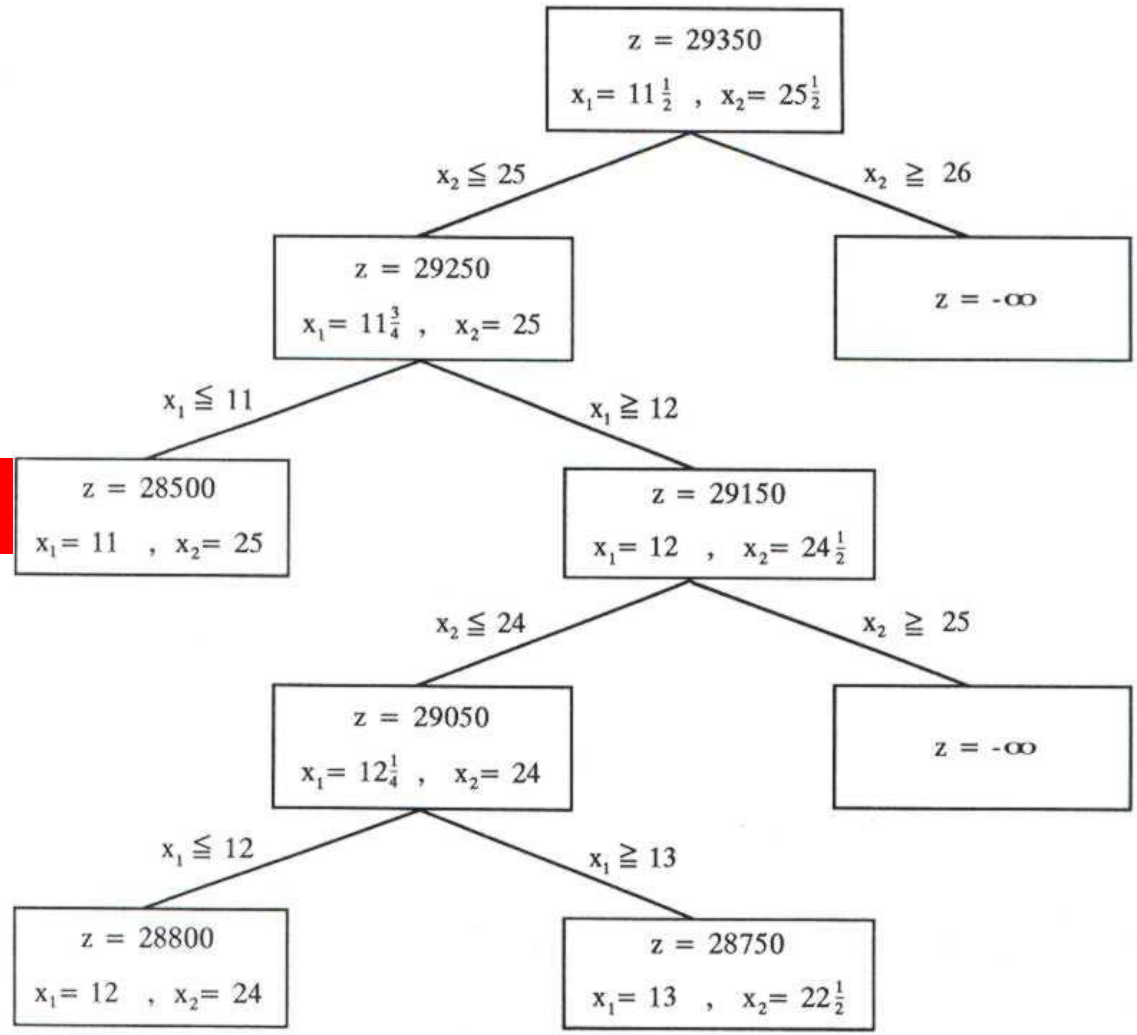
maximizar $1000x_1 + 700x$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros

LB = 28500



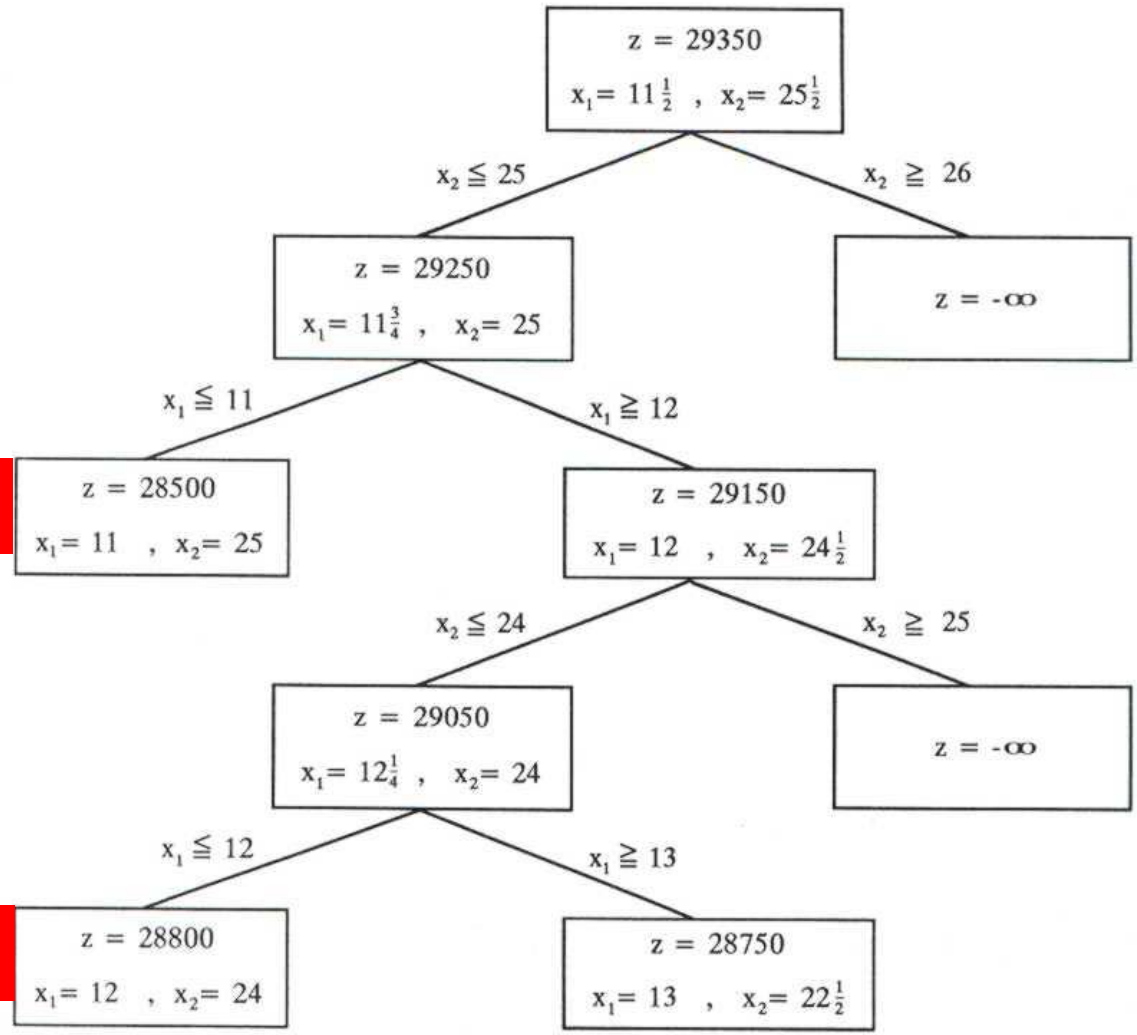
Métodos de solução

maximizar $1000x_1 + 700x$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros



LB = 28500

LB = 28800

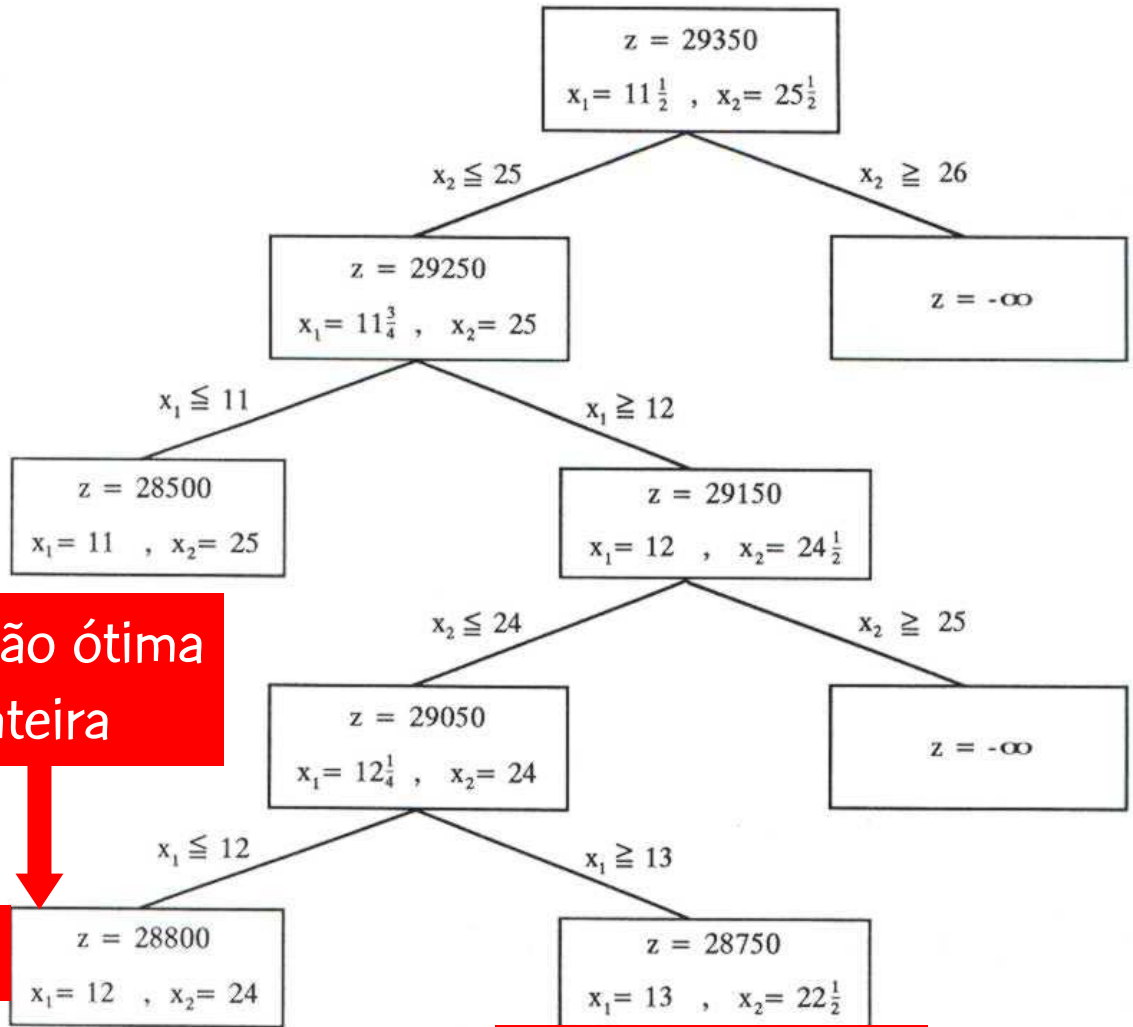
Métodos de solução

maximizar $1000x_1 + 700x$

sujeito a : $20x_2 \leq 510$

$100x_1 + 50x_2 \leq 2425$

$x_1, x_2 \geq 0$ e inteiros



Solução ótima inteira

LB = 28800

UB < LB ⇒ fim

Métodos de solução

- Exemplo 3: problema da mochila

maximizar $5x_1 + 3x_2 + 6x_3 + 6x_4 + 2x_5$

sujeito a : $5x_1 + 4x_2 + 7x_3 + 6x_4 + 2x_5 \leq 15$

$x_1, x_2, x_3, x_4, x_5 \in \{0,1\}$

Resolução da relaxação linear:

$$c_1 / a_1 = c_4 / a_4 = c_5 / a_5 > c_3 / a_3 > c_2 / a_2$$

Branch-and-bound em problemas 0-1:
fixar as variáveis em zero (esquerda) ou um (direita)

Métodos de solução

$z = 14\frac{5}{7}$

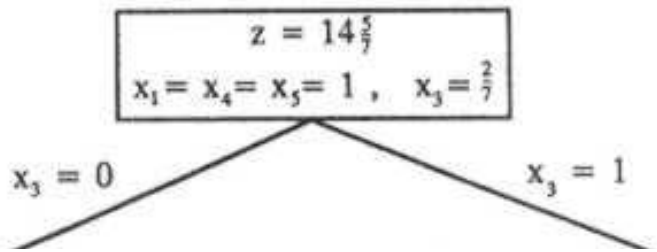
$x_1 = x_4 = x_5 = 1\,, \quad x_3 = \frac{2}{7}$

maximizar $5x_1 + 3x_2 + 6x_3 + 6x_4 + 2x_5$

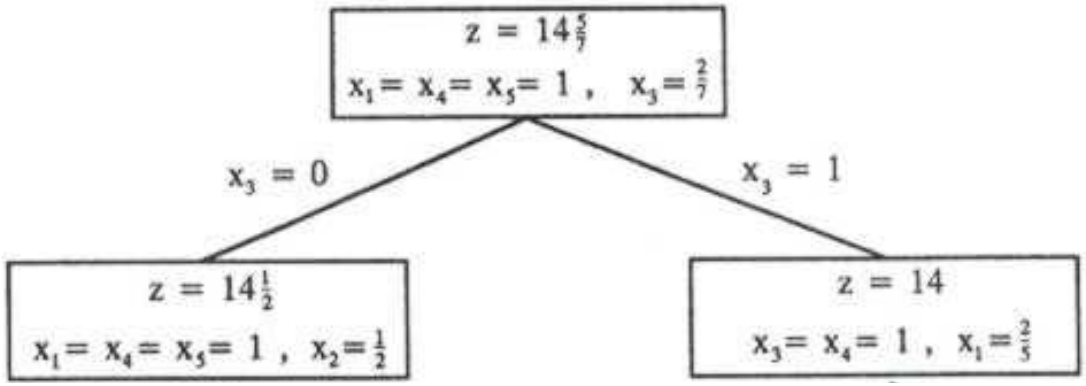
sujeito a : $5x_1 + 4x_2 + 7x_3 + 6x_4 + 2x_5 \leq 15$

$x_1, x_2, x_3, x_4, x_5 \in \{0,1\}$

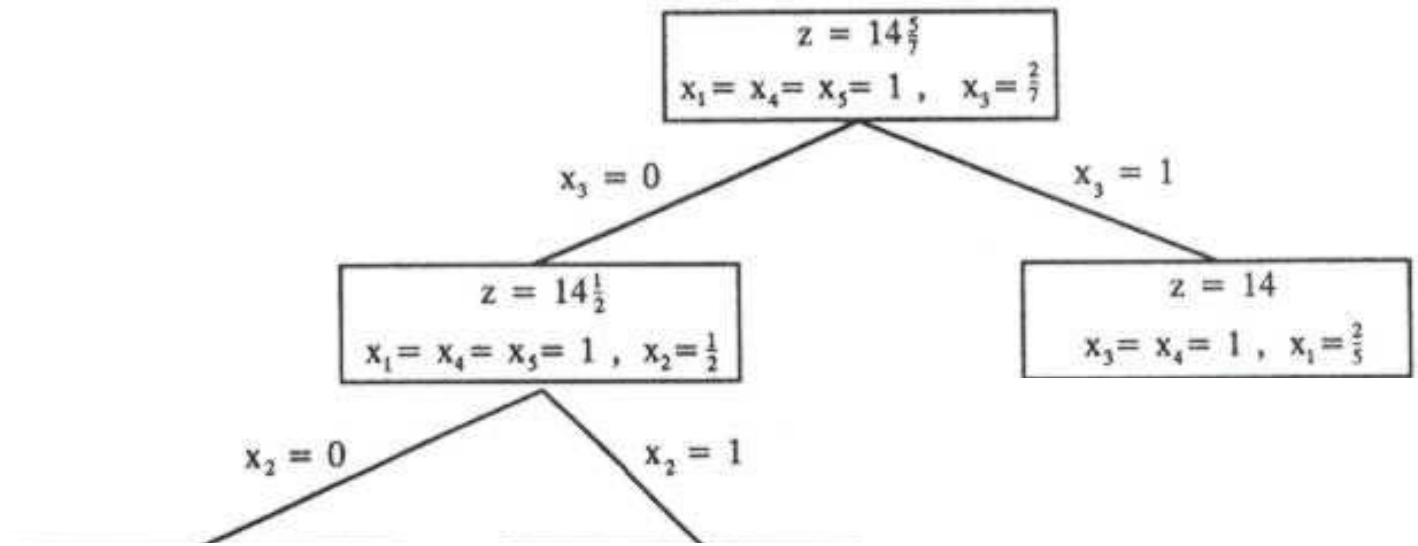
Métodos de solução



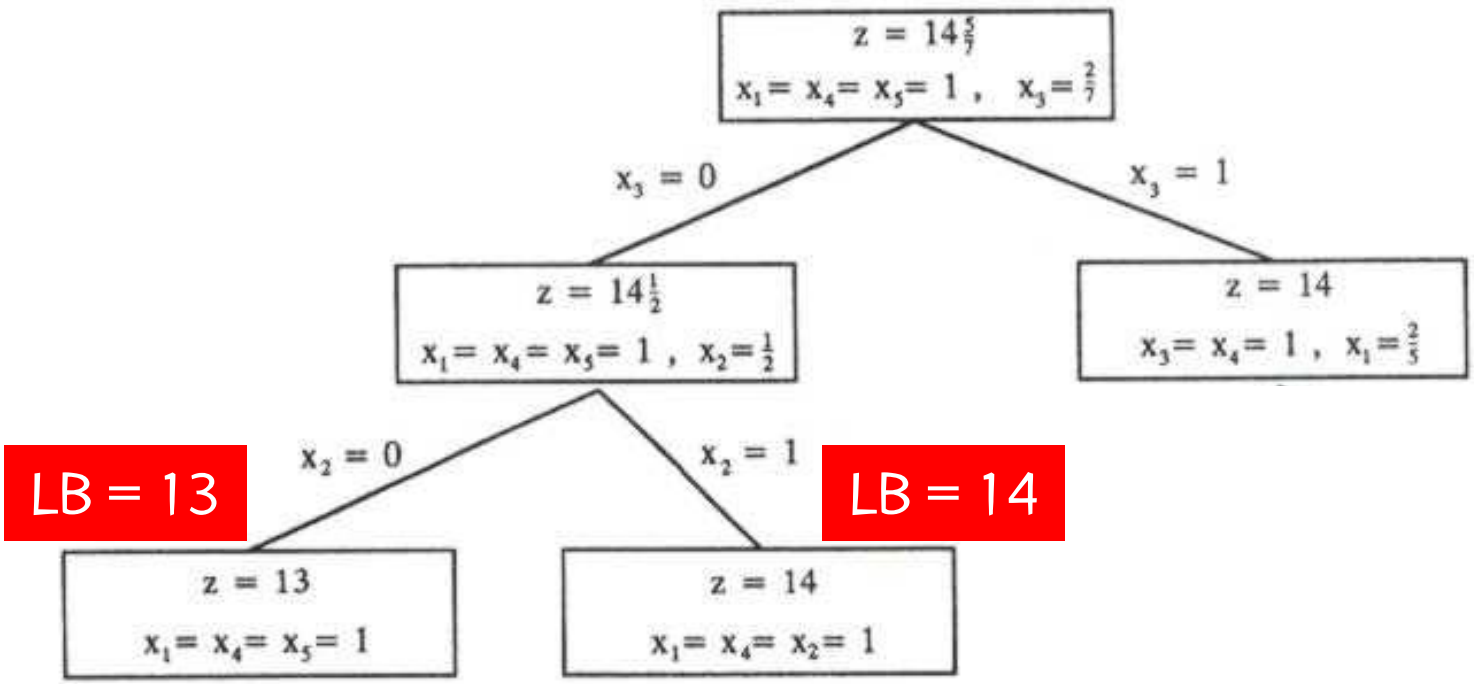
Métodos de solução



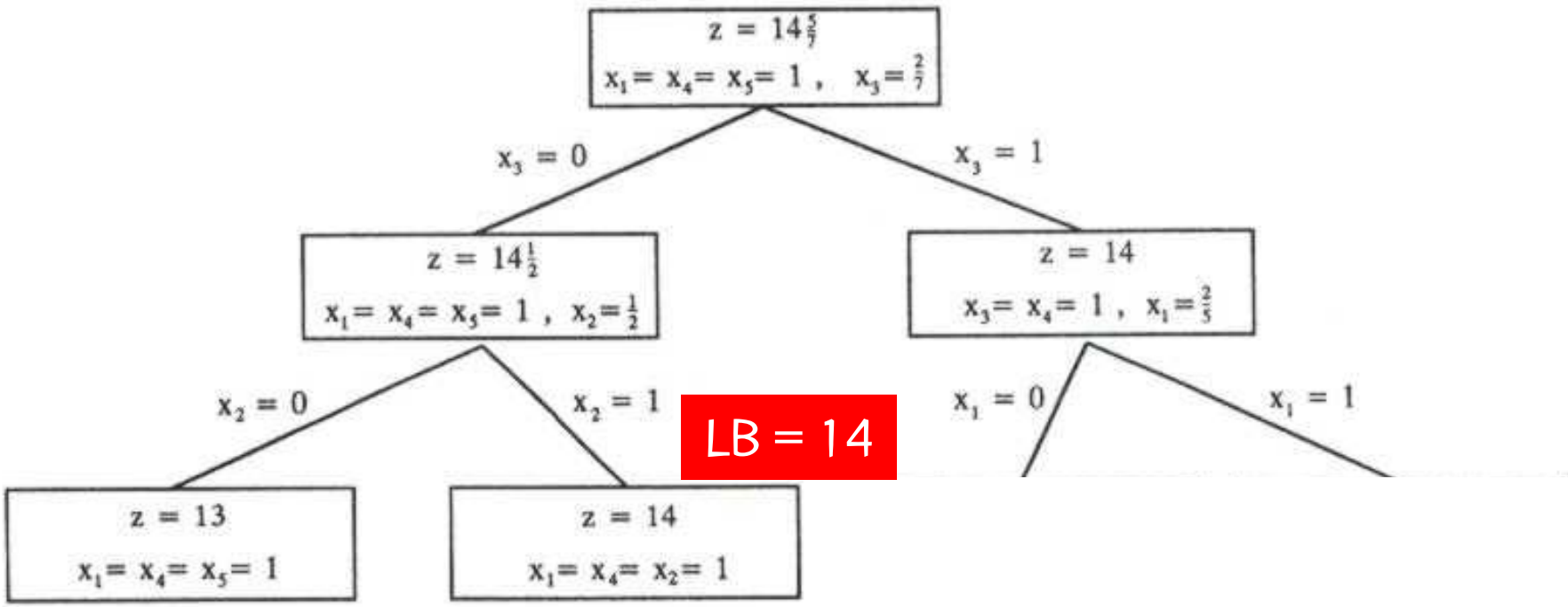
Métodos de solução



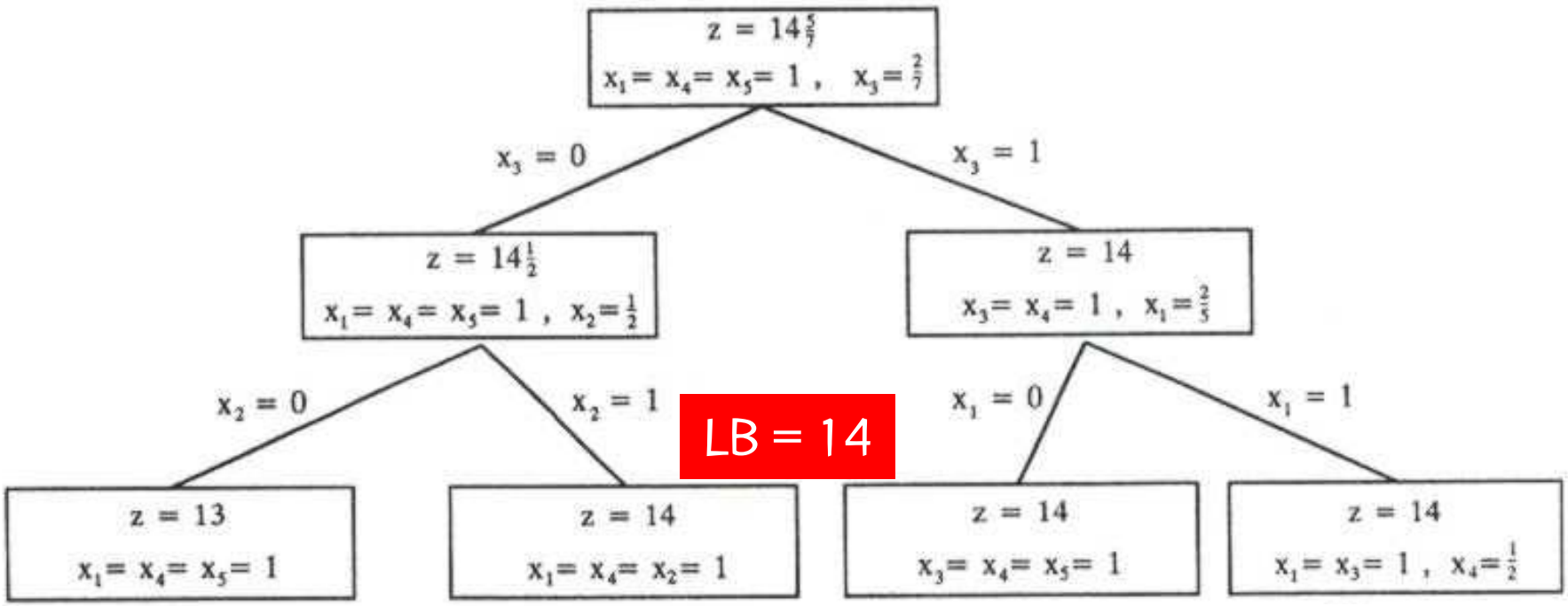
Métodos de solução



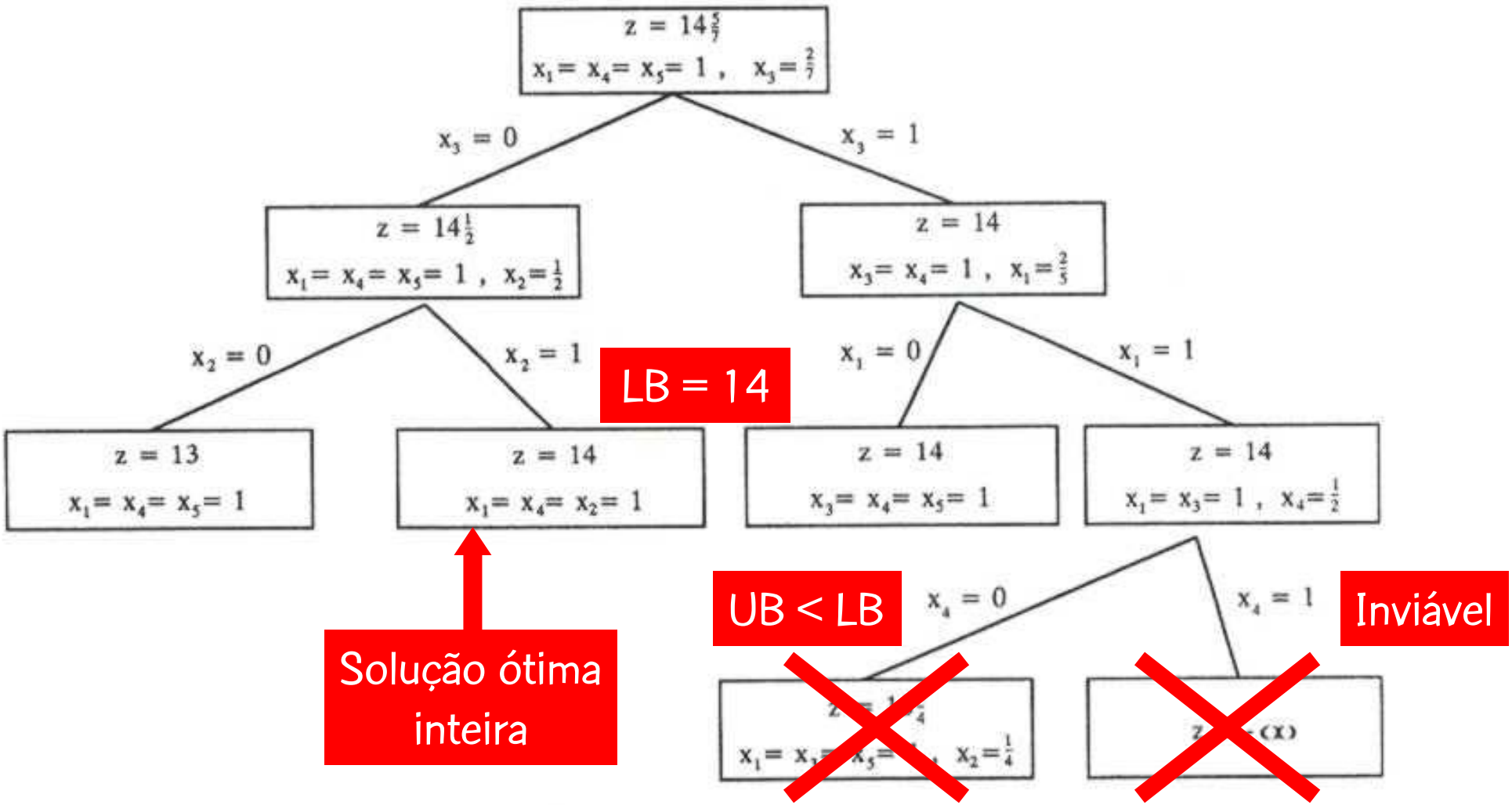
Métodos de solução



Métodos de solução



Métodos de solução



Métodos de solução

- Método de branch-and-bound também pode ser aplicado na solução de problemas de programação mista, onde nem todas as variáveis devem ser inteiras.
- Técnica de solução mais empregada para a solução de problemas de programação inteira.
- Extensões:
  - Branch-and-price
  - Branch-and-cut
  - Relax-and-cut, ...

Teoria da complexidade

- Por que os problemas de programação inteira são difíceis?
- A dificuldade dos problemas de programação inteira é intrínseca a eles ou é apenas devida a ainda não conhecermos bons algoritmos?
- Será possível que existam bons algoritmos genéricos para problemas de programação inteira?
- Quais são os problemas difíceis de serem resolvidos?
- Teoria da complexidade responde a algumas destas questões.

Teoria da complexidade

- Problemas de otimização: "dentre todas as estruturas satisfazendo determinada propriedade, obter aquela que otimiza certa função de custo."
  Resultado: uma solução viável ótima
- Problemas de decisão: "existe uma determinada estrutura satisfazendo determinada propriedade?"
  Resultado: "sim" ou "não"
- Resolver um problema de decisão: responder SIM ou NÃO

Teoria da complexidade

- Exemplo: problema do caixeiro viajante
  Entrada: n cidades e distâncias c_{ij}

  Problema de decisão: "dado um inteiro K, existe um ciclo hamiltoniano de comprimento menor ou igual a K?"

  Problema de otimização: "obter um ciclo hamiltoniano de comprimento mínimo."

Teoria da complexidade

- Exemplo: problema da mochila
  Entrada: n itens, peso máximo b, lucros c_j e pesos a_j associados a cada item j=1,...,n

  Problema de decisão: "dado um inteiro K, existe um subconjunto $S \subseteq \{1,...,n\}$ tal que $\sum_{j \in S} a_j \leq b$ e $\sum_{j \in S} c_j \geq K$?"

  Problema de otimização: "obter um conjunto S* maximizando $\sum_{j \in S} c_j$ entre todos os conjuntos $S \subseteq \{1,...,n\}$ tais que $\sum_{j \in S} a_j \leq b$."

Teoria da complexidade

- Exemplo: problema de programação linear
  Entrada: vetor c de coeficientes da função objetivo, coeficientes da matriz de restrições A, vetor b de coeficientes do segundo membro das restrições.

  Problema de decisão: "dado um inteiro K, existe uma solução $x \geq 0$ tal que $A.x = b$ e $c.x \leq K$?"

  Problema de otimização: "obter uma solução $x^* \geq 0$ que minimize o produto c.x e tal que $A.x^* = b$."

Teoria da complexidade

- Suponha-se conhecido um algoritmo para resolver o problema de otimização.
- Então, torna-se trivial resolver o problema de decisão a ele associado.
- Conseqüência: resolver o problema de decisão nunca é mais difícil do que resolver o problema de otimização.
- O estudo da teoria da complexidade é baseado nos problemas de decisão.

Teoria da complexidade

- De forma simplificada, define-se a complexidade de um algoritmo como uma medida de seu tempo de execução (número de operações elementares, número de instruções, número de linhas de código executadas, etc.) em função do tamanho de sua entrada de dados (L).
- Se a complexidade de um algoritmo cresce de acordo com a função f(L), diz-se que sua complexidade é O(f(L)) (isto é, diz-se que sua complexidade é da ordem de f(L)).

Teoria da complexidade

- Exemplos: considere-se um vetor de n elementos inteiros
  - Determinar o menor ou maior elemento: O(n)
  - Busca seqüencial por um elemento específico: O(n)
  - Busca binária por um elemento (vetor ordenado): O(log n)
  - Ordenar o vetor por um método "simples": $O(n^2)$
  - Ordenar o vetor por um método "elaborado": O(n log n)
- Exemplos: considere-se duas matrizes quadradas n x n
  - Calcular a soma das duas matrizes: $O(n^2)$
  - Calcular o produto das duas matrizes: $O(n^3)$

Teoria da complexidade

- Exemplos: considere um número inteiro n
  - Determinar se n é primo: O(n)
  - Determinar se n é primo (aprimorado): O(sqrt(n))
  - Calcular o fatorial de n: O(n!)
- Um algoritmo (determinístico) para um problema de decisão é polinomial se sua complexidade de pior caso é dada (ou limitada superiormente) por um polinômio no tamanho de sua entrada.
- Algoritmos polinomiais são considerados como bons ou eficientes.

Teoria da complexidade

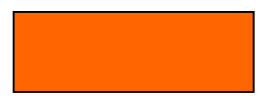
- Classe P: problemas de decisão para os quais são conhecidos algoritmos polinomiais para sua resolução.
- Resolver um problema de decisão: responder SIM ou NÃO
- Exemplos de problemas da classe P:
  - ordenação
  - caminhos mais curtos em um grafo, fluxo máximo em um grafo
  - programação linear
    - O algoritmo Simplex tem um excelente desempenho na prática, mas não é um algoritmo polinomial no pior caso (é possível criar instâncias onde ele explora um número exponencial de pontos extremos).
    - Entretanto, existem algoritmos polinomiais para programação linear (e.g., métodos de pontos interiores).

Teoria da complexidade

- Visão simplificada do "mundo" dos problemas de decisão:

Teoria da complexidade

- Classe NP: problemas de decisão para os quais são conhecidos algoritmos polinomiais que permitem verificar em tempo polinomial se uma resposta fornecida por um "oráculo" permite concluir pela resposta "sim".
- Resolver um problema de decisão: responder SIM ou NÃO
- Para os problemas da classe NP, não se exige a existência de um algoritmo resolvedor polinomial, mas sim apenas a existência de um algoritmo verificador (de respostas "sim") polinomial.



Teoria da complexidade

- A classe NP pode ser definida alternativamente através do conceito de algoritmos não-determinísticos.
- Algoritmos não-determinísticos são aqueles que fazem uso de uma instrução especial de desvio "duplo":

GO TO Label_A,Label_B

- Esta instrução funciona como se criasse duas cópias do fluxo de execução em um ambiente de paralelismo ilimitado, cada uma executando a partir de um ponto diferente.

Teoria da complexidade

- Exemplo: algoritmo não-determinístico para o problema da mochila (dado um inteiro K, existe um subconjunto S dos n objetos tal que $\sum_{j\in S} a_j \leq b$ e $\sum_{j\in S} c_j \geq K$?)

```
for  j = 1 to n by 1
     go to A,B
A:   x_j ← 0
     go to C
B:   x_j ← 1
C:   continue
if a.x ≤ b and c.x ≥ L then "sim"
```

Criação de 2^n cópias do vetor solução x, todas diferentes e cada uma delas associada a uma forma diferente de escolher objetos dentre os n disponíveis

Testa se alguma solução satisfaz às condições acima e leva à resposta "sim".

Teoria da complexidade

- Um algoritmo não-determinístico para um problema de decisão é polinomial se a complexidade de pior caso do primeiro ramo a responder "sim" é dada (ou limitada superiormente) por um polinômio no tamanho de sua entrada.
- Classe NP: problemas de decisão para os quais são conhecidos algoritmos não-determinísticos polinomiais para sua resolução.
- Resolver um problema de decisão: responder SIM ou NÃO

Teoria da complexidade

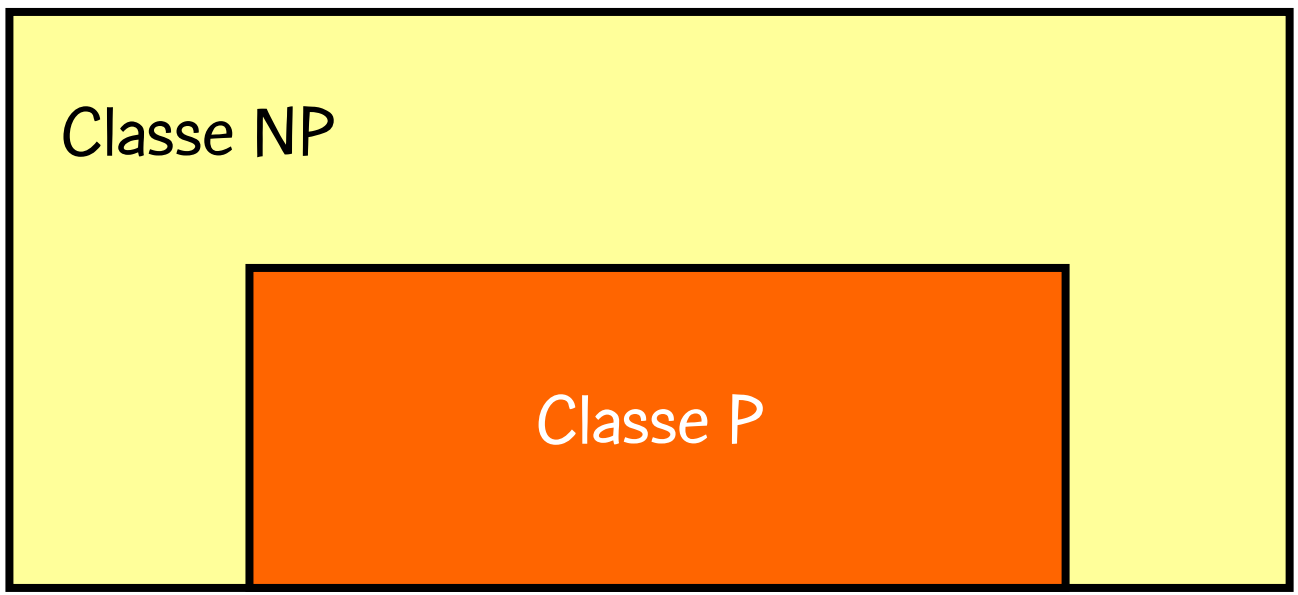
- É simples concluir que se existe um algoritmo resolvedor (polinomial) para um problema de decisão, então certamente existe um algoritmo verificador (polinomial) para este problema.
- Conseqüência: $P \subseteq NP$
- Questão em aberto: $P = NP$ ou $P \subset NP$?
  Será que há problemas para os quais não existem algoritmos resolvedores polinomiais (eficientes)?
- Esta é uma das questões em aberto mais importantes em Teoria da Computação.

Teoria da complexidade

- Visão simplificada do "mundo" dos problemas de decisão:

Teoria da complexidade

- Visão simplificada do "mundo" dos problemas de decisão:



Teoria da complexidade

- Transformação polinomial: um problema de decisão A se transforma polinomialmente em outro problema de decisão B se, dada uma instância do problema A, pode-se construir uma instância do problema B (em tempo polinomial no tamanho da instância A), tal que a resposta para a instância de A é "sim" se e somente se a resposta para a instância transformada de B também é "sim".

Teoria da complexidade

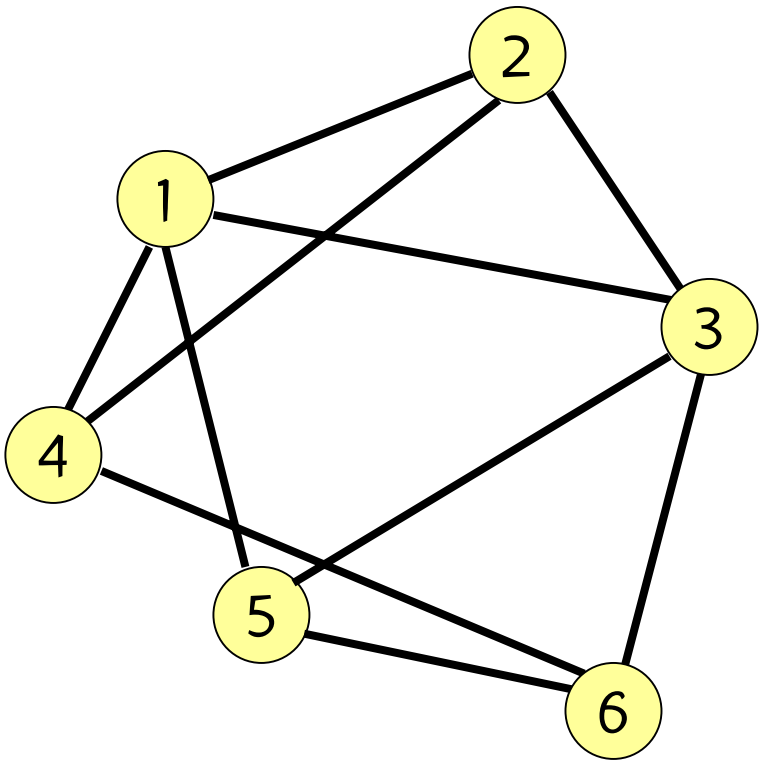
- Exemplo: clique se transforma em conjunto estável

  Problema da clique máxima: dado um grafo, determinar se existe uma clique (isto é, um subconjunto de nós onde todos os pares estão conectados diretamente por uma aresta) com pelo menos K nós.

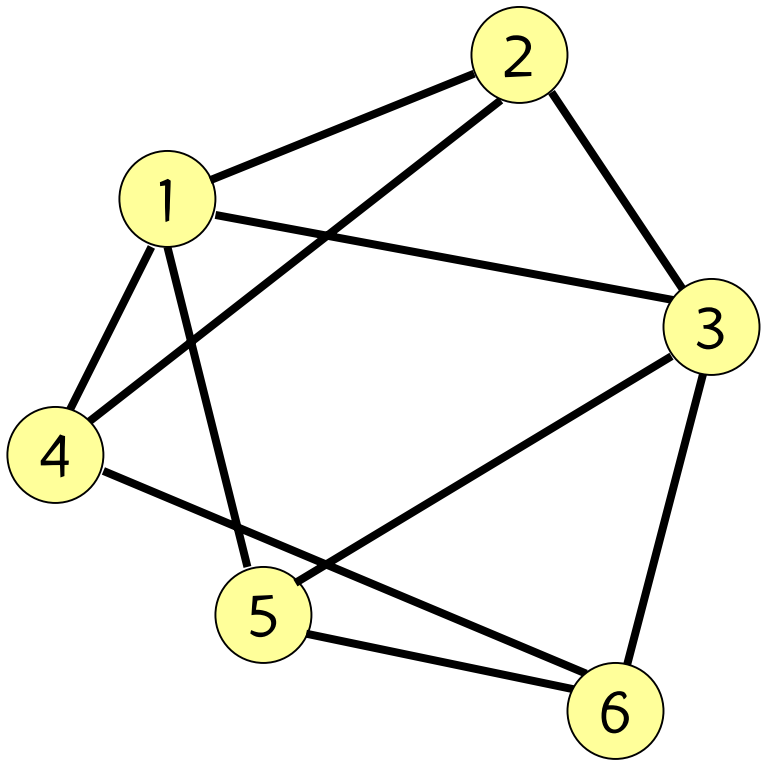
  Problema do conjunto estável máximo: dado um grafo, determinar se existe um conjunto estável (isto é, um subconjunto de nós onde nenhum par está conectado diretamente por uma aresta) com pelo menos K nós.

Teoria da complexidade

Grafo *G*



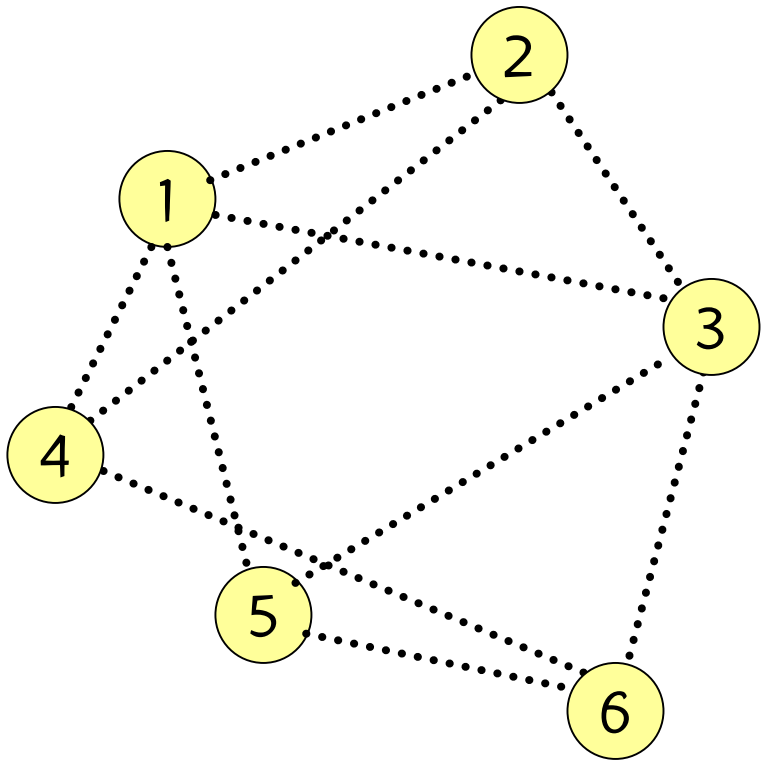
Teoria da complexidade



Grafo G

Grafo complementar de G: (i,j) é aresta do grafo complementar $\Leftrightarrow$ (i,j) não é aresta de G

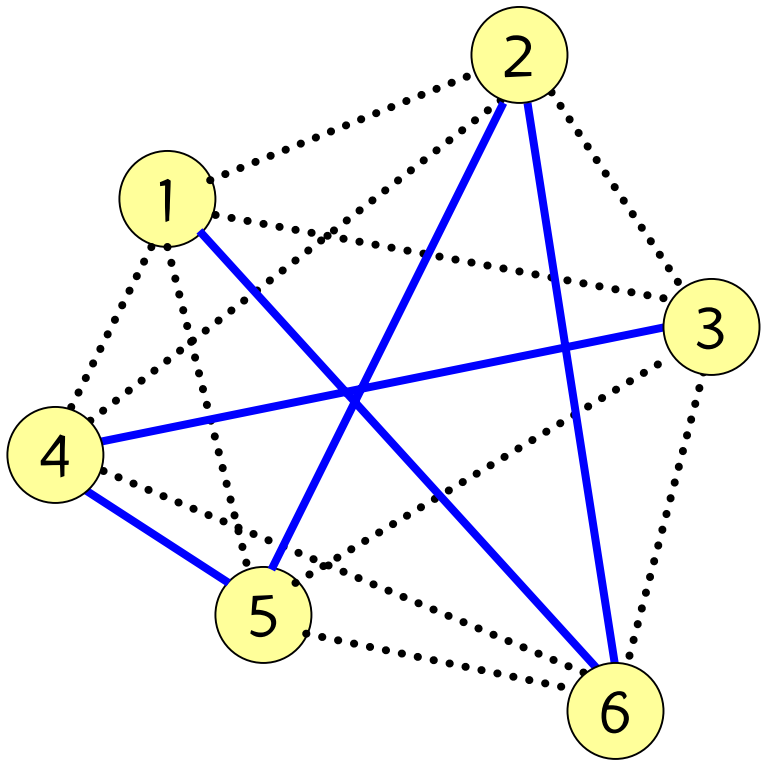
Teoria da complexidade



Grafo G

Grafo complementar de G: (i,j) é aresta do grafo complementar $\Leftrightarrow$ (i,j) não é aresta de G

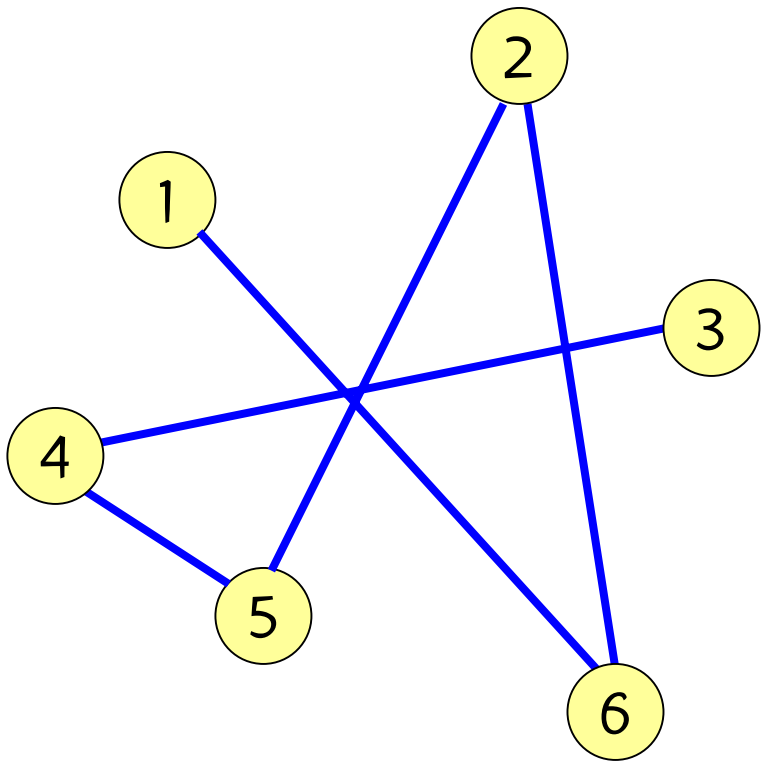
Teoria da complexidade



Grafo G

Grafo complementar de G: (i,j) é aresta do grafo complementar $\Leftrightarrow$ (i,j) não é aresta de G

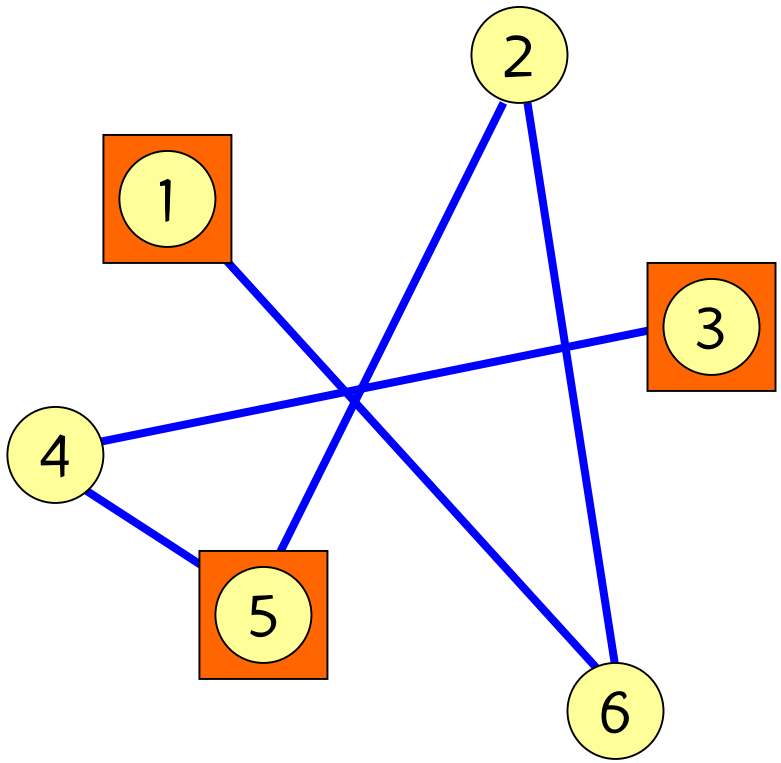
Teoria da complexidade



Grafo G

Grafo complementar de G: (i,j) é aresta do grafo complementar $\Leftrightarrow$ (i,j) não é aresta de G

Teoria da complexidade

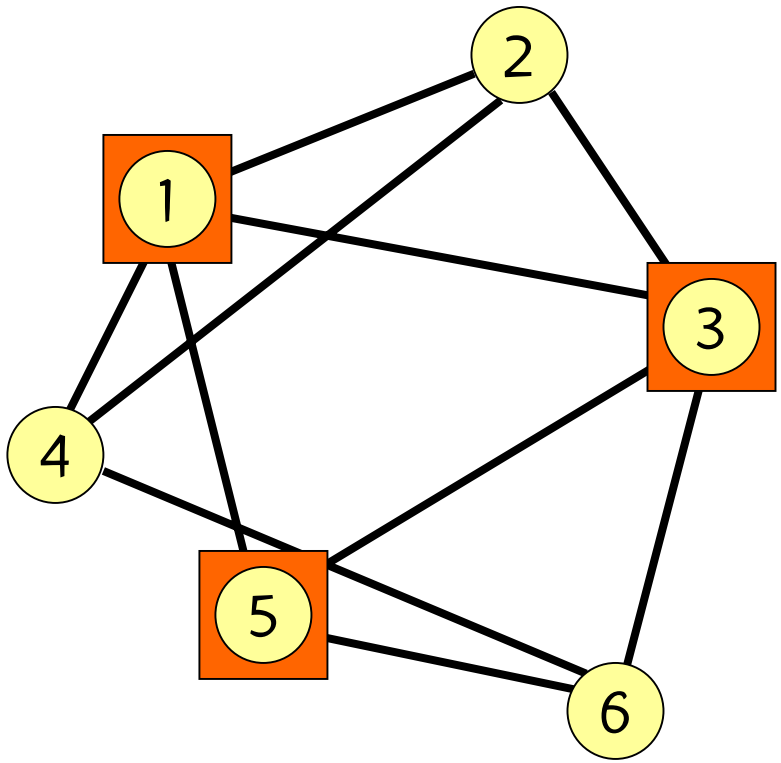


Grafo G

Grafo complementar de G: (i,j) é aresta do grafo complementar $\Leftrightarrow$ (i,j) não é aresta de G

Conjunto estável no grafo complementar: {1,3,5}

Teoria da complexidade



Grafo G

Grafo complementar de G: (i,j) é aresta do grafo complementar $\Leftrightarrow$ (i,j) não é aresta de G

Conjunto estável no grafo complementar: {1,3,5}

Clique no grafo G: {1,3,5}

Teoria da complexidade

- Suponha-se conhecido um algoritmo para resolver o problema da clique máxima.
- Como resolver o problema do conjunto estável máximo?
- Simples!
  - Construir o grafo complementar.
  - Aplicar o algoritmo para resolver o problema da clique máxima no grafo complementar.
  - A clique máxima no grafo complementar é igual ao conjunto estável máximo no grafo original.

Teoria da complexidade

- Há transformações deste tipo entre muitos problemas:
  - Coloração de grafos $\leftrightarrow$ clique máxima
  - Clique máxima $\leftrightarrow$ caixeiro viajante
  - Caixeiro viajante $\leftrightarrow$ problema da mochila
  - etc.
- Problemas NP-completos: diz-se que um problema de decisão da classe NP é NP-completo se todos os outros problemas desta classe se transformam polinomialmente nele.

Teoria da complexidade

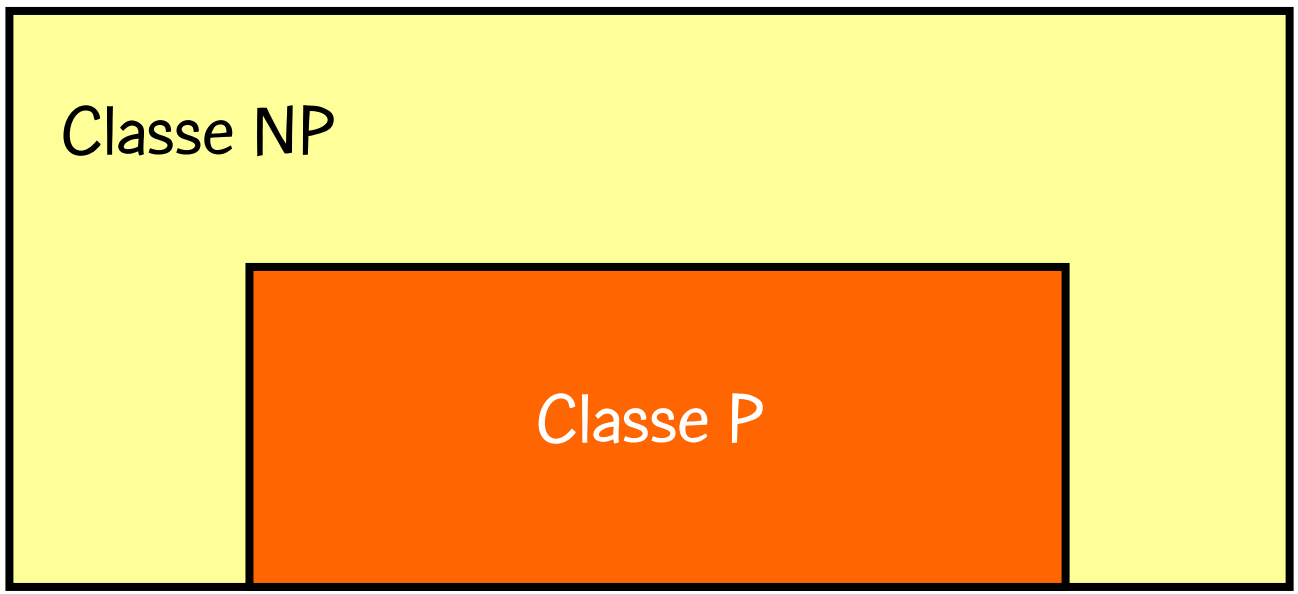
- Conseqüência: se existir um algoritmo (determinístico) polinomial para resolver algum problema NP-completo, então todos os problemas da classe NP também poderão ser resolvidos em tempo polinomial.
- Por esta razão, os problemas NP-completos podem ser considerados como os mais difíceis da classe NP.
- Não existem ou ainda não foram encontrados algoritmos polinomiais (eficientes) para os problemas NP-completos.

Teoria da complexidade

- Exemplos de problemas NP-completos:
  - Caixeiro viajante
  - Problema da mochila
  - Coloração de grafos
  - Particionamento
  - Recobrimento
  - p-medianas
  - Localização de facilidades
  - Programação inteira

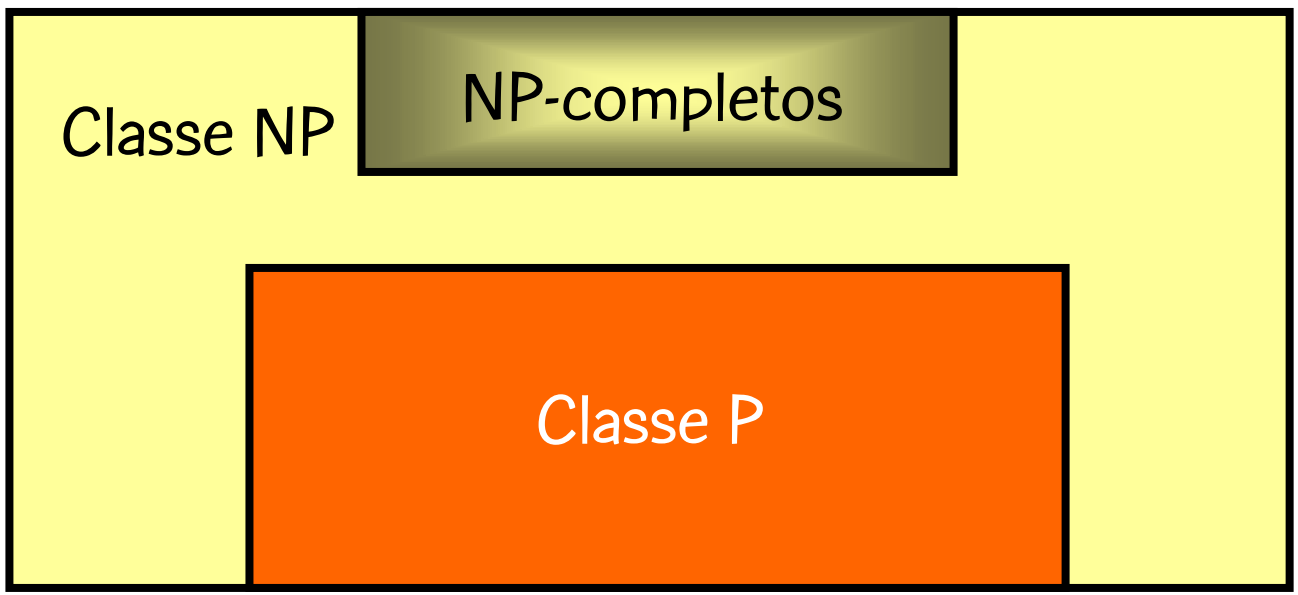
Teoria da complexidade

- Visão simplificada do "mundo" dos problemas de decisão:



Teoria da complexidade

- Visão simplificada do "mundo" dos problemas de decisão:



Teoria da complexidade

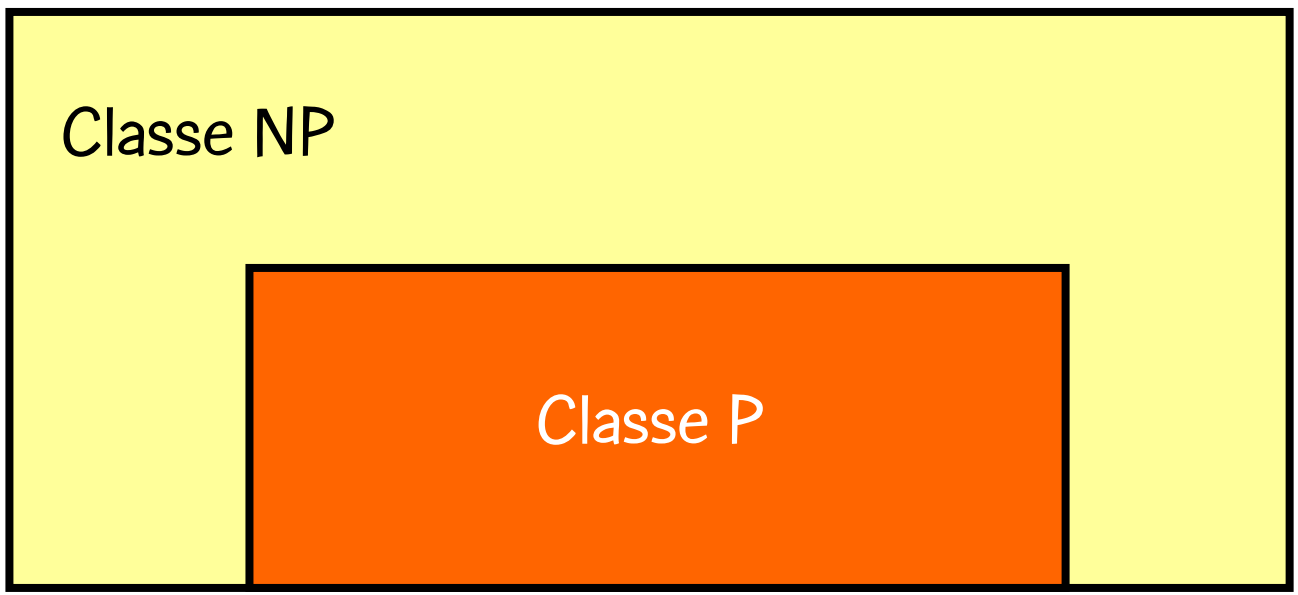
- Classe PSPACE: problemas de decisão que podem ser resolvidos de forma eficiente utilizando uma quantidade polinomial de memória
  - $P \subseteq PSPACE$
  - $NP \subseteq PSPACE$

Teoria da complexidade

- Visão simplificada do "mundo" dos problemas de decisão:

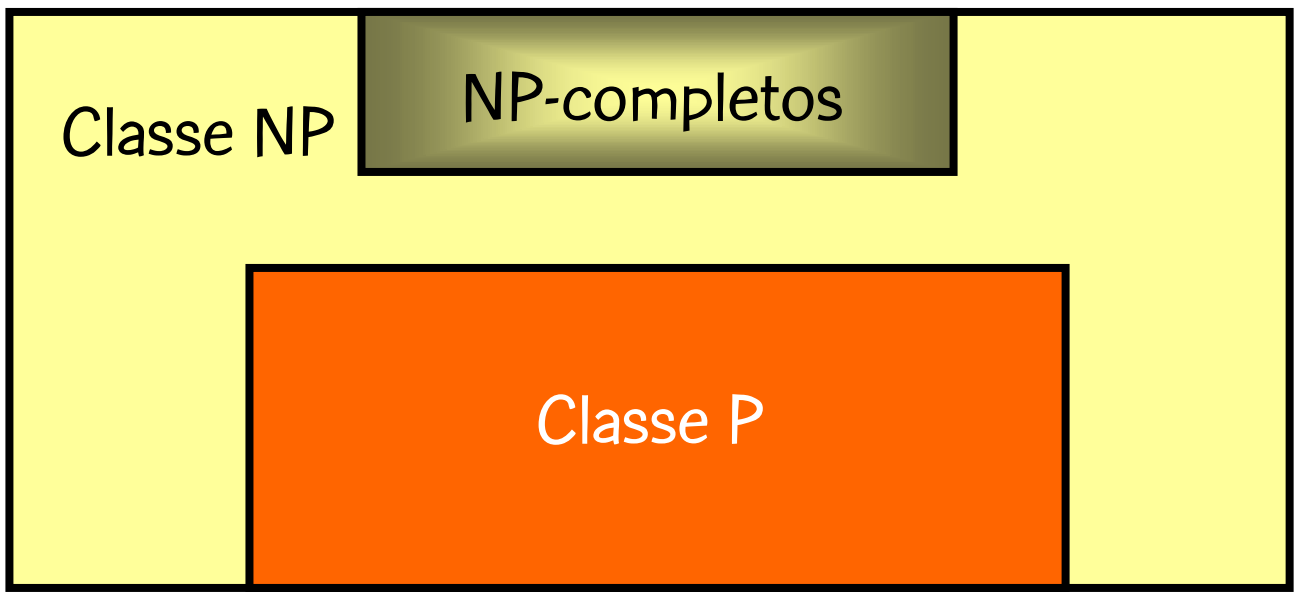
Teoria da complexidade

- Visão simplificada do "mundo" dos problemas de decisão:



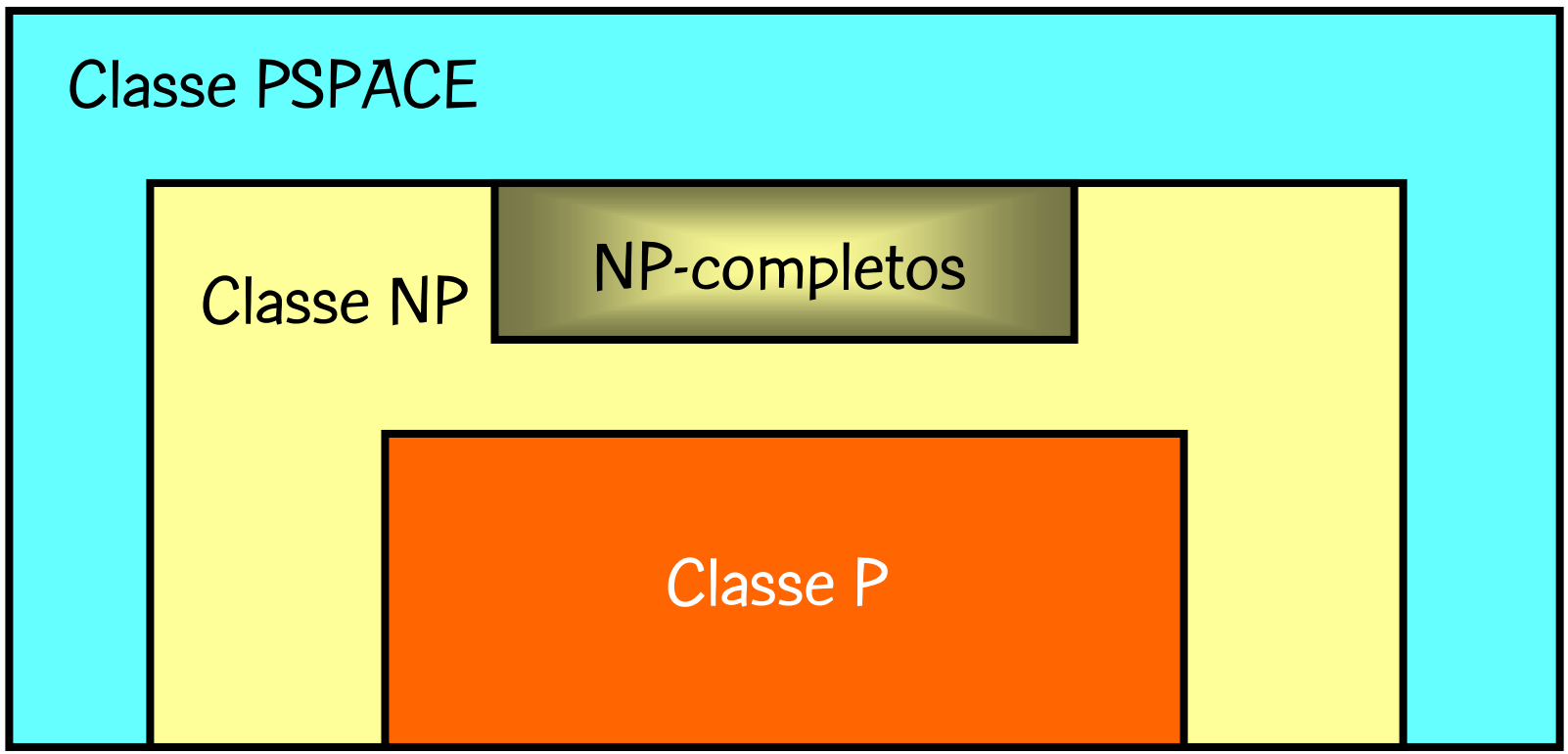
Teoria da complexidade

- Visão simplificada do "mundo" dos problemas de decisão:



Teoria da complexidade

- Visão simplificada do "mundo" dos problemas de decisão:



Teoria da complexidade

- Como tratar na prática os problemas NP-completos?

1. Métodos exatos de complexidade super-polinomial (branch-and-bound, cortes, enumeração, programação dinâmica): se for necessário e se o porte dos problemas assim o permitir.
2. Processamento paralelo: clusters e grids permitem acelerações significativas na prática, utilizando recursos computacionais básicos e com custo reduzido.

Teoria da complexidade

- Como tratar na prática os problemas NP-completos?

3. Heurísticas: qualquer método aproximado projetado com base nas propriedades estruturais ou nas características das soluções dos problemas, com complexidade reduzida em relação à dos algoritmos exatos e fornecendo, em geral, soluções viáveis de boa qualidade (sem garantia de qualidade)
   - Métodos construtivos
   - Busca local
   - Metaheurísticas

Teoria da complexidade

- Avanços no estudo e desenvolvimento de heurísticas:
  - Resolver problemas maiores
  - Resolver problemas em tempos menores
  - Obter melhores soluções